(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平8－263290

(43)公開日　平成８年(1996)10月11日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　9/38	３７０		Ｇ０６Ｆ　9/38	３７０Ｂ	
	３８０			３８０Ｃ	

審査請求　未請求　請求項の数18　ＯＬ　（全 73 頁）

(21)出願番号　特願平7－60797

(22)出願日　平成７年(1995)３月20日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(71)出願人　390023928
日立エンジニアリング株式会社
茨城県日立市幸町３丁目２番１号

(72)発明者　榊原　栄二
東京都小平市上水本町５丁目20番１号　株式会社日立製作所半導体事業部内

(72)発明者　三ツ石　直幹
東京都小平市上水本町５丁目20番１号　株式会社日立製作所半導体事業部内

(74)代理人　弁理士　秋田　収喜

最終頁に続く

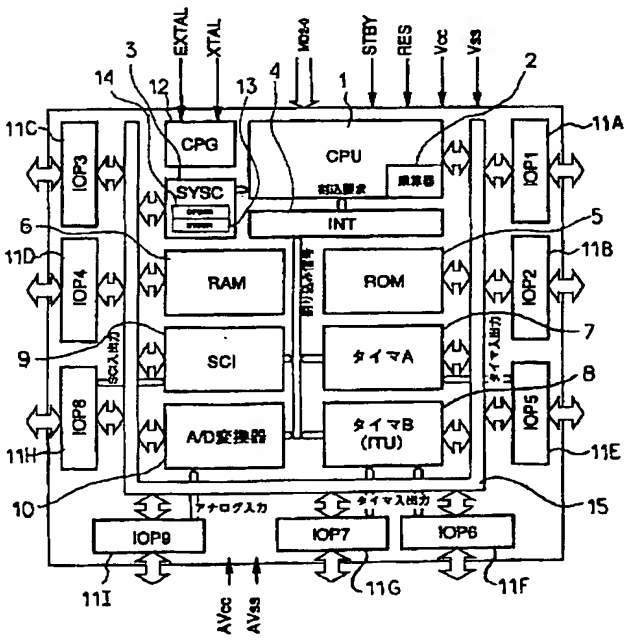
(54)【発明の名称】　データ処理装置

(57)【要約】

【目的】　互換性を維持しつつニーズに応じた乗算性能が得られ、しかも処理性能の向上を図ることが可能な技術を提供する。

【構成】　命令を実行する実行手段を制御する制御手段と乗算手段を設け、制御手段と乗算手段とが並列動作する第１の命令とともに、前記乗算手段が動作する第２の命令を有し、乗算手段を前記制御手段に内蔵するように構成する。第１の命令は積和命令であるとともに、第２の命令は乗算命令になっている。

図１



1

【特許請求の範囲】

【請求項１】　所定の命令を順次実行するデータ処理装置であって、前記命令を実行する実行手段を制御する制御手段と乗算手段を設け、前記制御手段と乗算手段とが並列動作する第１の命令とともに、前記乗算手段が動作する第２の命令を有することを特徴とするデータ処理装置。

【請求項２】　前記第２の命令が第１の命令と同期していることを特徴とする請求項２に記載のデータ処理装置。

【請求項３】　前記第１の命令は積和命令であるとともに、前記第２の命令は乗算命令であることを特徴とする請求項１または２に記載のデータ処理装置。

【請求項４】　前記乗算手段は前記制御手段に内蔵されていることを特徴とする請求項１乃至３のいずれか１項に記載のデータ処理装置。

【請求項５】　前記乗算手段は前記制御手段から独立して設けられていることを特徴とする請求項１乃至３のいずれか１項に記載のデータ処理装置。

【請求項６】　前記制御手段は乗算機能を備えていることを特徴とする請求項５に記載のデータ処理装置。

【請求項７】　前記乗算手段の動作の有効／無効を指定できる手段を有することを特徴とする請求項１乃至６のいずれか１項に記載のデータ処理装置。

【請求項８】　前記乗算手段の動作の有効／無効を指定できる手段をテストモードで設定することを特徴とする請求項７に記載のデータ処理装置。

【請求項９】　前記乗算手段の動作の有効／無効を指定できる手段をエミュレーションモードで設定することを特徴とする請求項７に記載のデータ処理装置。

【請求項１０】　前記乗算手段の動作の有効／無効を指定できる手段がレジスタからなり、このレジスタはエミュレータに搭載されて、エミュレータの指定に基づいて設定することを特徴とする請求項９に記載のデータ処理装置。

【請求項１１】　前記乗算手段の１ステップ動作の実行を指定する手段を有することを特徴とする請求項を１乃至１０に記載のデータ処理装置。

【請求項１２】　前記乗算手段の１ステップ動作の実行を指定する手段をテストモードで設定することを特徴とする請求項１１に記載のデータ処理装置。

【請求項１３】　所定の命令を順次実行するデータ処理装置であって、前記命令を実行する実行手段を制御する制御手段に指定可能な複数のレジスタの組み合わせを固定にし、複数のレジスタの退避／復帰命令を有することを特徴とするデータ処理装置。

【請求項１４】　所定の命令を順次実行するデータ処理装置であって、前記命令を実行する実行手段を制御する制御手段に搭載されるコントロールレジスタの有効／無効を切り換える手段を有することを特徴とするデータ処

2

理装置。

【請求項１５】前記コントロールレジスタの有効時には、例外処理の遷移時、例外処理からの復帰時に、前記コントロールレジスタの待避／復帰を行い、前記コントロールレジスタの無効時には、例外処理の遷移時、例外処理からの復帰時に、前記コントロールレジスタの待避／復帰を行わないことを特徴とする請求項１４に記載のデータ処理装置。

【請求項１６】　前記コントロールレジスタの有効時には、このコントロールレジスタの設定値が有効であり、前記コントロールレジスタの無効時には、このコントロールレジスタの設定値が無視されることを特徴とする請求項１５に記載のデータ処理装置。

【請求項１７】　所定の命令を順次実行するデータ処理装置であって、前記命令を実行する実行手段を制御する制御手段の搭載される固定的なスタックレジスタを設け、エミュレーションプログラムへの遷移時、エミュレーションプログラムからの復帰時に、前記固定的なスタックレジスタを用いて、ユーザが使用するスタックポインタを無視あるいは保持する手段を有することを特徴とするデータ処理装置。

【請求項１８】　前記エミュレーションプログラム実行状態での、エミュレーション遷移割り込みを禁止する手段を有することを特徴とする請求項１７に記載のデータ処理装置

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、データ処理装置に関し、特に、半導体集積回路装置によって構成される高速かつ小型のシングルチップマイクロコンピュータに利用して有効な技術に関するものである。

【０００２】

【従来の技術】半導体集積回路装置の製造技術の高度化に伴って、半導体単結晶からなるシングルチップに、中央演算処理装置（Ｃｅｎｔｒａｌ　Ｐｒｏｃｅｓｓｉｎｇ　Ｕｎｉｔ；以下、単にＣＰＵと称する）、プログラムを格納するＲＯＭ（ＲｅａｄＯｎｌｙ　Ｍｅｍｏｒｙ）、書き替え可能に各種データを格納するＲＡＭ（Ｒａｎｄｏｍ　Ａｃｃｅｓｓ　Ｍｅｍｏｒｙ）等を含む構成素子を集積して製造した、小型のシングルチップマイクロコンピュータ（以下、単にマイクロコンピュータと称する）が広範囲に普及してきており、種々の目的のデータ処理装置として使用されてきている。このマイクロコンピュータは、ＣＰＵが同時に処理し得る情報の量によって性能が異なり、例えば４ビット、８ビット、１６ビット、３２ビット等のマイクロコンピュータとして区分されている。

【０００３】このようなマイクロコンピュータは、アドレス空間の拡張や、命令セットの拡大、高速化等が図られてきている。また、ＣＰＵは、ソフトウエアによって

その性能が定義されているから、前記のようにアドレス空間の拡張や、命令セットの拡大、高速化等を図ったマイクロコンピュータにおいても、既存のマイクロコンピュータのソフトウエア資産を有効に利用できることが望ましい。

【０００４】このため、オブジェクトレベルで互換性を保ちつつ、アドレス空間の拡張や、命令セットの拡大、高速化等を実現した例として、例えば本出願人が先に提案した特開平６－５１９８１号公報、あるいは平成５年６月（株）日立製作所発行、「Ｈ８／３００Ｈシリーズプログラミングマニュアル」等がある。

【０００５】前記ＣＰＵは、システムクロックの２周期である、いわゆる２ステートで基本命令を実行している。これに対して、１ステートで基本命令を実行するようにし、さらに、ＣＰＵとは独立して乗算器を内蔵して高速化を図った例として、例えば平成４年１１月日経ＢＰ社発行、「日経エレクトロニクスＮＯ．５６８」、ＰＰ９９～ＰＰ１１２、あるいは平成５年３月（株）日立製作所発行、「ＳＨ７０３２、ＳＨ７０３４　ハードウエアマニュアル」等がある。このような乗算器は積和演算と乗算に利用する。

【０００６】このように高速化を図ることによって、マイクロコンピュータによって制御される各種機器の高速化や高性能化、あるいは、従来においては複数の半導体集積回路装置で構成していたものを、結合したりすることにより小型化を図ることができるようになる。

【０００７】また、前記のような各種機器の高速化や高性能化、あるいは小型化は、アドレス空間が比較的小さく、命令セットが比較的小さいＣＰＵあるいはマイクロコンピュータにおいても要求されるから、前記特開平６－５１９８１号公報等に記載されるアドレス空間の広いＣＰＵと、アドレス空間の小さいＣＰＵが存在する場合には、その双方の高速化を図ることが望ましい。

【０００８】このような観点から、上位ＣＰＵを開発し、これをベースにして下位ＣＰＵへ展開できれば都合が良い。これによって、開発効率を向上することができる。さらに、半導体集積回路装置によって構成されるＣＰＵ自体の他に、クロスアセンブラやＣコンパイラ、シミュレータ、リアルタイムＯＳ等の開発ツール等の開発も共通化して、開発効率を向上することが望ましい。

【０００９】

【発明が解決しようとする課題】前記のようなマイクロコンピュータにおいて、乗算器は専用の資源を必要とするから、必ずしも積和演算や乗算の高速化を必要としない場合には、費用対効果の点で得策でない。また、例えば前記平成５年３月（株）日立製作所発行、「ＳＨ７０３２、ＳＨ７０３４　ハードウエアマニュアル」においては、乗算結果は専用のレジスタ（ＭＡＣ）に得られるから、これを利用する場合には、別の命令によってそれをＣＰＵの汎用レジスタに転送しなければならない。乗算器を内蔵して乗算自体を高速化しても、そのように乗算結果を使用するまでの時間が長くなっては意味がない。

【００１０】一方、従来のＣＰＵとの互換性を維持するためには、前記のように命令の追加は困難であり、追加する命令は最小限にしなければならない。また、演算結果等のフラグも互換性を保持する必要がある。積和演算についても、演算結果等のフラグを参照できれば使い勝手が良くなる。フラグの状態を判定して分岐する、いわゆる条件分岐命令などで演算結果を容易に判定し、処理の内容を変更することができるからである。かかるフラグには、オーバフロー（Ｖ）、ゼロ（Ｚ）、ネガティブ（Ｎ）などがある。

【００１１】本発明の目的は、互換性を維持しつつニーズに応じた乗算性能が得られ、しかも処理性能の向上を図ることが可能な技術を提供することにある。

【００１２】本発明の他の目的は、制御手段に乗算手段を内蔵して互換性を維持しつつ処理の高速化を図ることが可能な技術を提供することにある。

【００１３】本発明のその他の目的は、乗算手段を制御手段から独立して設け、しかも制御手段に乗算機能を備えさせることにより、製造費用の低減が可能な技術を提供することにある。

【００１４】本発明の前記ならびにそのほかの目的と新規な特長は、本発明書の記述および添付図面から明らかになるであろう。

【００１５】

【課題を解決するための手段】本願において開示される発明のうち代表的なものの概要を簡単に説明すれば下記の通りである。

【００１６】（１）本発明のデータ処理装置は、命令を実行する実行手段を制御する制御手段と乗算手段を設け、前記制御手段と乗算手段とが並列動作する第１の命令とともに、前記乗算手段が動作する第２の命令を有し、乗算手段は前記制御手段に内蔵されている。また、第１の命令は積和命令であるとともに、第２の命令は乗算命令になっている。積和命令のアドレッシングモードは、いわゆるポストインクリメントレジスタ間接とする。乗算手段には結果を判定するフラグ検出手段を設け、乗算命令時はフラグ検出結果を制御手段に供給して、保持させる手段を設ける。

【００１７】（２）本発明のデータ処理装置は、命令を実行する実行手段を制御する制御手段と乗算手段を設け、前記制御手段と乗算手段とが並列動作する第１の命令とともに、前記乗算手段が動作する第２の命令を有し、乗算手段は前記制御手段から独立して設けられている。また、制御手段は乗算機能を備えている。

【００１８】（３）本発明のデータ処理装置は、命令を実行する実行手段を制御する制御手段に指定可能な複数のレジスタの組み合わせを固定にし、複数のレジスタの

退避／復帰命令を有している。

【００１９】（４）本発明のデータ処理装置は、命令を実行する実行手段を制御する制御手段に搭載されるコントロールレジスタの有効／無効を切り換える手段を有し、コントロールレジスタの有効時には、例外処理の遷移時、例外処理からの復帰時に、前記コントロールレジスタの待避／復帰を行い、前記コントロールレジスタの無効時には、例外処理の遷移時、例外処理からの復帰時に、前記コントロールレジスタの待避／復帰を行わない。

【００２０】（５）本発明のデータ処理装置は、命令を実行する実行手段を制御する制御手段の搭載される固定的なスタックレジスタを設け、エミュレーションプログラムへの遷移時、エミュレーションプログラムからの復帰時に、前記固定的なスタックレジスタ用いて、ユーザが使用するスタックポインタを無視あるいは保持するかを指定する手段を有している。

【００２１】

【作用】上記した（１）の手段によれば、乗算器（乗算手段）を内蔵することによって、アドレッシングモードの増加を最小限にして、かつ処理性能を低下させずに積和演算を実行可能にすることができる。また、ポストインクリメントレジスタ間接により、多数のデータの積和演算を連続して処理することができる。さらに、乗算の結果（積、フラグ）を直ちに利用できるから、実質的な乗算の実行速度を向上することができる。

【００２２】乗算器とＣＰＵ（制御手段）を一体に構成して、乗算器・ＣＰＵ間の配線を短縮して、物理的規模を縮小する。また、高速化に寄与することができる。

【００２３】上記した（２）の手段によれば、乗算器を取外し可能に（独立して）設けることによって、乗算器を取外した場合は、積和演算をサポートしないことによって、容易に下位ＣＰＵを実現し、論理的・物理的規模を縮小し、製造費用を低減した別のマイクロコンピュータを容易に開発することができる。また、乗算器を取外したＣＰＵにおいても、汎用的な乗算命令をサポートすることによって、使い勝手の低下を防止できる。さらに、乗算器使用するか使用しないかの制御信号（有効／無効）を与えて制御することによって、テスト性を向上したり、エミュレータを共通化したりすることができる。さらにまた、全体的な開発効率を向上することができる。

【００２４】乗算器を削除した場合、乗算は除算と同一のシーケンスで実行できる。積和演算はサポートせず、積和演算の特殊なシーケンスをサポートしないことによって論理規模の縮小を更に行うことができる。テスト命令をサポートすることによって、論理規模の増加を最低限にして、テストの容易性を向上することができる。

【００２５】上記した（３）の手段によれば、複数レジスタの退避／復帰命令を持ち、この組み合せを固定的にすることによって、論理規模の縮小を図ることができ、また、高速化を図ることができる。レジスタの本数の異なる命令を複数命令サポートすることによって、使い勝手の低下を防ぐことができる。

【００２６】さらに、内部動作のパイプラインに対応して、入出力タイミングの異なるレジスタ選択回路を複数持つことにより、レジスタ間演算命令などの基本命令を実質的に１命令／１ステート実行を行うことができる。

【００２７】上記した（４）の手段によれば、コントロールレジスタの有効／無効を切り換えることで、スタックの節約と、割込み応答時間の高速化に寄与することができる。また、互換性を維持することができる。

【００２８】上記した（５）の手段によれば、エミュレータ専用の固定スタックポインタを持つことにより、エミュレータのサポートを容易にすることができる。また、論理規模の増加を最低限にして、エミュレータの設計を容易にすることができる。エミュレータ専用スタックポインタの一部のアドレスを、ＣＰＵ外部から与えるようにして、スタックレジスタをリロケータブルにし、マイクロコンピュータのアドレス配置などに容易に対応することができる。

【００２９】以下、本発明について、図面を参照して実施例とともに詳細に説明する。

【００３０】なお、実施例を説明するための全図において、同一機能を有するものは同一符号を付け、その繰り返しの説明は省略する。

【００３１】

【実施例】図１に、本発明の適用されたデータ処理装置の一例であるシングルチップマイクロコンピュータ（以下、単にマイクロコンピュータと称する）のブロック図を示す。マイクロコンピュータは、ＣＰＵ１、乗算器２、システムコントローラ（ＳＹＳＣ）３、割込コントローラ（ＩＮＴ）４、ＲＯＭ５、ＲＡＭ６、タイマＡ７、タイマＢ８、シリアルコミュニケーションインタフェース（ＳＣＩ）９、Ａ／Ｄ変換器１０、第１乃至第９入出力ポート（ＩＯＰ１～ＩＯＰ９）１１Ａ～１１Ｉ、クロック発振器（ＣＰＧ）１２の機能ブロック乃至はモジュールから構成され、公知の半導体製造技術により１つの半導体基板上に半導体集積回路装置として形成される。ＣＰＵ１は、乗算器２を内蔵してなる。システムコントローラ（ＳＹＳＣ）３は、システムコントロールレジスタ（ＳＹＳＣＲ）１３および制御レジスタ（ＣＰＵＣＲ）１４を内蔵している。

【００３２】かかるマイクロコンピュータは、電源端子として、グランドレベル（Ｖｓｓ）、電源電圧レベル（Ｖｃｃ）、その他専用制御端子として、リセット（ＲＥＳ）、スタンバイ（ＳＴＢＹ）、モード制御（ＭＤ０～２）、クロック入力（ＥＸＴＡＬ、ＸＴＡＬ）端子を有する。クロック入力（ＥＸＴＡＬ、ＸＴＡＬ）端子に接続される、図示はされない水晶振動子に基づいて、ク

ロック発振器が生成するシステムクロック（φ１、φ２）に同期して、マイクロコンピュータは動作する。或は外部クロックをＥＸＴＡＬ端子に入力してもよい。システムクロックの１周期を１ステートと呼ぶ。

【００３３】これらの機能ブロックは、内部バスによって相互に接続される。内部バスは内部アドレスバス（ＰＡＢ）・内部データバス（ＰＤＢ）の他、リード信号・ライト信号を含み、さらにバスサイズ信号或いはシステムクロック（φ１、φ２）などを含む。

【００３４】入出力ポートは、外部バス信号、入出力回路の入出力信号と兼用とされている。これらは、動作モードあるいはソフトウエアの設定により、機能を選択されて、使用される。ＩＯＰ１～３はアドレスバス出力、ＩＯＰ４、５はデータバス入出力、ＩＯＰ６はバス制御信号入出力信号と兼用されている。外部アドレスは、それぞれ、これらの入出力ポートに含まれるバッファ回路を介して内部アドレスバスと接続されている。

【００３５】内部バスおよび外部バス共に１６ビットバス幅とし、バイトサイズ（８ビット）およびワードサイズ（１６ビット）のリード／ライトを可能にする。なお、内部バスおよび外部バスのいずれも８ビット幅とすることもできる。バス制御信号入出力信号には、アドレスストローブ信号ＡＳ、リード信号ＲＤ、ライト信号ＨＷＲ・ＬＷＲ、ウェイト信号ＷＡＩＴ、エリア０選択信号ＣＳ０などがある。割込信号は、タイマ・ＳＣＩ・ＩＯＰ８から要求され、割込コントローラ（ＩＮＴ）が調停して、ＣＰＵに割込を要求する。このとき、ＣＰＵに対し、割込要求信号とベクタ番号を与える。

【００３６】ＲＥＳ端子にリセット信号が加えられると、モード端子（ＭＤ０～２）で与えられる動作モードを取り込み、マイクロコンピュータはリセット状態になる。モード端子で設定する動作モードは、シングルチップ／拡張、アドレス空間、内蔵ＲＯＭの有効／無効、データバス幅の初期値を８ビットまたは１６ビットから選択する。

【００３７】図２に、システムコントロールレジスタ（ＳＹＳＣＲ）３の構成を示す。各ビットの内容を表１乃至表４に示す。

【００３８】なお、ビット２、１：リザーブビット　リードすると常に”０”が読み出される。ライトは無効である。

【００３９】

【表１】

表１

ビット７：MACサチュレーション（MACS）

MAC命令の飽和演算、非飽和演算を選択します。

ビット７ MACS	説明
0	MAC命令は非飽和演算（初期値）
1	MAC命令は飽和演算

【００４０】

【表２】

表２

ビット５、４：割込み制御モード１、０（INTM1、0）

割込みコントローラの割込み制御モードを選択します。

ビット５ INTM1	ビット４ INTM0	割込み制御モード	説明
0	0	0	Iビットで、割込みを制御します。（初期値）
0	1	1	I、UIビットで、割込みを制御します。
1	0	2	I2～I0ビットで、割込みを制御します。
1	1	3	I、UI、I2～I0ビットで、割込みを制御します。

【００４１】

【表３】

9　　10

表３

ビット３：ＮＭＩエッジセレクト（ＮＭＩＥＧ）

NMI割込みの入力エッジ選択を行ないます。

ビット３ NMIEG	説明
0	NMI入力の立下がりエッジで割込み要求を発生（初期値）
1	NMI入力の立上がりエッジで割込み要求を発生

【００４２】

【表４】

表４

ビット０：RAMイネーブル

内蔵RAMの有効または無効を選択します。RAMEビットはリセット状態の解除時に初期化されます。ソフトウェアスタンバイモードでは初期化されません。

ビット０ RAME	説明
0	内蔵RAM無効
1	内蔵RAM有効（初期値）

【００４３】以下に、表５にＣＰＵ１の命令セットを示す。本実施例に用いられるＣＰＵ１の命令は合計で７１種類ある。表６に命令とアドレッシングモードとの組合せを示す。表７に以下の各表に使用される記号（オペレーションの記号）の意味を示す。表８乃至表１５に各命令の機能別一覧表を示す。

【００４４】

【表５】

11　　12

表５

命令の分類

分類	命令	サイズ	種類
データ転送命令	MOV	BWL	7
	POP,PUSH	WL	
	LDM，STM	L	
	MOVFPE,MOVTPE	B	
算術演算命令	ADD,SUB,CMP,NEG	BWL	23
	ADDX,SUBX,DAA,DAS	B	
	INC,DEC	BWL	
	ADDS,SUBS	L	
	MULXU,DIVXU,MULXS,DIVXS	BW	
	EXTU,EXTS	WL	
	TAS	B	
	MAC,LDMAC,STMAC,CLRMAC	-	
論理演算命令	AND,OR,XOR,NOT	BWL	4
シフト命令	SHAL,SHAR,SHLL,SHLR,ROTL,ROTR,ROTXL,ROTXR	BWL	8
ビット操作命令	BSET,BCLR,BNOT,BTST,BLD,BILD,BST,BIST BAND,BIAND,BOR,BIOR,BXOR,BIXOR	B	14
分岐命令	Bcc,JMP,BSR,JSR,RTS	-	5
システム制御命令	TRAPA,RTE,SLEEP,LDC,STC,ANDC,ORC,XORC,NOP	-	9
ブロック転送命令	EEPMOV	-	1

合計７１種類

Ｂ：バイトサイズ　Ｗ：ワードサイズ　Ｌ：ロングワードサイズ

*1　POP.W　Rn、PUSH.W　Rnは、それぞれMOV.W　@SP+,Rn、MOV.W　Rn,@-SPと同一です。
また、POP.L　ERn、PUSH.L　ERnは、それぞれMOV.L　@SP+,ERn、MOV.L　ERn,@-SPと同一です。

*2　Bccは条件分岐命令の総称です。

30

【００４５】

【表６】

40

50

13　　14

表6

命令とアドレッシングモードの組合せ

機能	命令	アドレッシングモード													
		#xx	Rn	@ERn	@(d:16,ERn)	@(d:32,ERn)	@-ERn/@ERn+	@aa:8	@aa:16	@aa:24	@aa:32	@(d:8,PC)	@(d:16,PC)	@@aa:8	—
データ転送命令	MOV	BWL	BWL	BWL	BWL	BWL	BWL	B	BWL	—	BWL	—	—	—	—
	POP、PUSH	—	—	—	—	—	—	—	—	—	—	—	—	—	WL
	LDM、STM	—	—	—	—	—	—	—	—	—	—	—	—	—	L
	MOVFPE、MOVTPE	—	—	—	—	—	—	—	B	—	—	—	—	—	—
算術演算命令	ADD、CMP	BWL	BWL	—	—	—	—	—	—	—	—	—	—	—	—
	SUB	WL	BWL	—	—	—	—	—	—	—	—	—	—	—	—
	ADDX、SUBX	B	B	—	—	—	—	—	—	—	—	—	—	—	—
	ADDS、SUBS	—	L	—	—	—	—	—	—	—	—	—	—	—	—
	INC、DEC	WL	BWL	—	—	—	—	—	—	—	—	—	—	—	—
	DAA、DAS	—	B	—	—	—	—	—	—	—	—	—	—	—	—
	MULXU、DIVXU	—	BW	—	—	—	—	—	—	—	—	—	—	—	—
	MULXS、DIVXS	—	BW	—	—	—	—	—	—	—	—	—	—	—	—
	NEG	—	BWL	—	—	—	—	—	—	—	—	—	—	—	—
	EXTU、EXTS	—	WL	—	—	—	—	—	—	—	—	—	—	—	—
	TAS	—	—	B	—	—	—	—	—	—	—	—	—	—	—
	MAC	—	—	—	—	—	○	—	—	—	—	—	—	—	—
	CLRMAC	—	—	—	—	—	—	—	—	—	—	—	—	—	○
	LDMAC、STMAC	—	L	—	—	—	—	—	—	—	—	—	—	—	—
論理演算命令	AND、OR、XOR	BWL	BWL	—	—	—	—	—	—	—	—	—	—	—	—
	NOT	—	BWL	—	—	—	—	—	—	—	—	—	—	—	—
シフト命令		—	BWL	—	—	—	—	—	—	—	—	—	—	—	—
ビット操作命令		—	B	B	—	—	—	B	B	—	B	—	—	—	—
分岐命令	Bcc、BSR	—	—	—	—	—	—	—	—	—	—	○	○	—	—
	JMP、JSR	—	—	—	—	—	—	—	—	○	—	—	—	○	—
	RTS	—	—	—	—	—	—	—	—	—	—	—	—	—	○
システム制御命令	TRAPA	—	—	—	—	—	—	—	—	—	—	—	—	—	○
	RTE	—	—	—	—	—	—	—	—	—	—	—	—	—	○
	SLEEP	—	—	—	—	—	—	—	—	—	—	—	—	—	○
	LDC	B	B	W	W	W	W	—	W	—	W	—	—	—	—
	STC	—	B	W	W	W	W	—	W	—	W	—	—	—	—
	ANDC、ORC、XORC	B	—	—	—	—	—	—	—	—	—	—	—	—	—
	NOP	—	—	—	—	—	—	—	—	—	—	—	—	—	○
ブロック転送命令		—	—	—	—	—	—	—	—	—	—	—	—	—	BW

【００４６】

【表７】

40

50

15　　16

表7

《オペレーションの記号》

Rd	汎用レジスタ(デスティネーション側)*
Rs	汎用レジスタ(ソース側)*
Rn	汎用レジスタ*
ERn	汎用レジスタ（32ビットレジスタ）
MAC	積和レジスタ（32ビットレジスタ）
(EAd)	デスティネーションオペランド
(EAs)	ソースオペランド
EXR	エクステンドレジスタ
CCR	コンディションコードレジスタ
N	CCRのN（ネガティブ）フラグ
Z	CCRのZ（ゼロ）フラグ
V	CCRのV（オーバフロー）フラグ
C	CCRのC（キャリ）フラグ
PC	プログラムカウンタ
SP	スタックポインタ
#IMM	イミディエイトデータ
disp	ディスプレースメント
+	加算
−	減算
×	乗算
÷	除算
∧	論理積
∨	論理和
⊕	排他的論理和
→	転送
～	反転論理(論理的補数)
:3／:8／:16／:24／:32	3／8／16／24／32ビット長

* 汎用レジスタは、8ビット（R0H～R7H,R0L～R7L）、16ビット（R0～R7,E0～E7）、または32ビットレジスタ（ER0～ER7）です。

30

【００４７】　　【表８】

40

50

17　　18

表８

命令の機能別一覧(1)

分類	命令	サイズ*	機　　能
データ転送命令	MOV	B／W／L	(EAs)→Rd、Rs→(EAd) 汎用レジスタと汎用レジスタ、または汎用レジスタとメモリ間でデータ転送します。また、イミディエイトデータを汎用レジスタに転送します。
	MOVFPE	B	(EAs)→Rd 外部メモリの内容(@aa:16で指定)をEクロックに同期したタイミングで汎用レジスタに転送します。
	MOVTPE	B	Rs→(EAs) 汎用レジスタの内容をEクロックに同期したタイミングで外部メモリ(@aa:16で指定)に転送します。
	POP	W／L	@SP+→Rn スタックから汎用レジスタへデータを復帰します。 POP.W RnはMOV.W @SP+,Rnと、また、 POP.L ERnはMOV.L @SP+,ERnと同一です。
	PUSH	W／L	Rn→@-SP 汎用レジスタの内容をスタックに退避します。 PUSH.W RnはMOV.W Rn,@-SPと、また、 PUSH.L ERnはMOV.L ERn,@-SPと同一です。
	LDM	L	@SP+→Rn（レジスタ群） スタックから複数の汎用レジスタへデータを復帰します。
	STM	L	Rn（レジスタ群）→@-SP 複数の汎用レジスタの内容をスタックに退避します。

【００４８】　　【表９】

30

40

50

表９

命令の機能別一覧(2)

分類	命　令	サイズ*	機　　能
算術演算命令	ADD SUB	B／W／L	Rd±Rs→Rd、Rd±#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の加減算を行います（バイトサイズでの汎用レジスタとイミディエイトデータ間の減算はできません。SUBX命令またはADD命令を使用してください）。
	ADDX SUBX	B	Rd±Rs±C→Rd、Rd±#IMM±C→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間のキャリ付きの加減算を行います。
	INC DEC	B／W／L	Rd±1→Rd、Rd±2→Rd 汎用レジスタに1または2を加減算します（バイトサイズで1の加減算のみ可能です）。
	ADDS SUBS	L	Rd±1→Rd、Rd±2→Rd、Rd±4→Rd 32ビットレジスタに1、2、または4を加減算します。
	DAA DAS	B	Rd(10進補正)→Rd 汎用レジスタ上の加減算結果をCCRを参照して4ビットBCDデータに補正します。
	MULXU	B／W	Rd×Rs→Rd 汎用レジスタと汎用レジスタ間の符号なし乗算を行います。 8ビット×8ビット→16ビット、16ビット×16ビット→32ビットの乗算が可能です。
	MULXS	B／W	Rd×Rs→Rd 汎用レジスタと汎用レジスタ間の符号付き乗算を行います。 8ビット×8ビット→16ビット、16ビット×16ビット→32ビットの乗算が可能です。
	DIVXU	B／W	Rd÷Rs→Rd 汎用レジスタと汎用レジスタ間の符号なし除算を行います。 16ビット÷8ビット→商8ビット余り8ビット、 32ビット÷16ビット→商16ビット余り16ビットの除算が可能です。
	DIVXS	B／W	Rd÷Rs→Rd 汎用レジスタと汎用レジスタ間の符号付き除算を行います。 16ビット÷8ビット→商8ビット余り8ビット、 32ビット÷16ビット→商16ビット余り16ビットの除算が可能です。
	CMP	B／W／L	Rd－Rs、Rd－#IMM 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の比較を行い、その結果をCCRに反映します。
	NEG	B／W／L	0－Rd→Rd 汎用レジスタの内容の2の補数(算術的補数)をとります。
	EXTU	W／L	Rd(ゼロ拡張)→Rd 16ビットレジスタの下位8ビットをワードサイズにゼロ拡張します。 または、32ビットレジスタの下位16ビットをロングワードサイズにゼロ拡張します。

【００４９】　【表１０】

表１０

命令の機能別一覧(3)

分類	命令	サイズ*	機　　能
算術演算命令	EXTS	W／L	Rd(符号拡張)→Rd 16ビットレジスタの下位8ビットをワードサイズに符号拡張します。または、32ビットレジスタの下位16ビットをロングワードサイズに符号拡張します。
	TAS	B	@ERd－0、(1)→(<ビット7>of @ERd) メモリの内容をテストした後、最上位ビット（ビット7）を"1"にセットします。
	MAC	—	(EAs)×(EAd)＋MAC→MAC メモリとメモリ間の符号付き乗算を行い、結果を積和レジスタに加算します。 16ビット×16ビット＋32ビット→32ビットの飽和演算、 16ビット×16ビット＋42ビット→42ビットの非飽和演算が可能です。
	CLRMAC	—	0→MAC 積和レジスタをゼロクリアします。
	LDMAC STMAC	L	Rs→MAC、MAC→Rd 汎用レジスタと積和レジスタ間でデータ転送します。
論理演算命令	AND	B／W／L	Rd∧Rs→Rd、Rd∧#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の論理積をとります。
	OR	B／W／L	Rd∨Rs→Rd、Rd∨#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の論理和をとります。
	XOR	B／W／L	Rd⊕Rs→Rd、Rd⊕#IMM→Rd 汎用レジスタと汎用レジスタ、または汎用レジスタとイミディエイトデータ間の排他的論理和をとります。
	NOT	B／W／L	～Rd→Rd 汎用レジスタの内容の1の補数(論理的補数)をとります。
シフト命令	SHAL SHAR	B／W／L	Rd(シフト処理)→Rd 汎用レジスタの内容を算術的にシフトします。
	SHLL SHLR	B／W／L	Rd(シフト処理)→Rd 汎用レジスタの内容を論理的にシフトします。
	ROTL ROTR	B／W／L	Rd(ローテート処理)→Rd 汎用レジスタの内容をローテートします。
	ROTXL ROTXR	B／W／L	Rd(ローテート処理)→Rd 汎用レジスタの内容をキャリフラグを含めてローテートします。

【００５０】　　【表１１】

23　　24

表１１

命令の機能別一覧(4)

分類	命令	サイズ*	機　　能
ビット操作命令	BSET	B	1→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された1ビットを"1"にセットします。ビット番号は、3ビットのイミディエイトデータまたは汎用レジスタの内容下位3ビットで指定します。
	BCLR	B	0→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された1ビットを"0"にクリアします。ビット番号は、3ビットのイミディエイトデータまたは汎用レジスタの内容下位3ビットで指定します。
	BNOT	B	～(<ビット番号>of<EAd>)→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転します。ビット番号は、3ビットのイミディエイトデータまたは汎用レジスタの内容下位3ビットで指定されます。
	BTST	B	～(<ビット番号>of<EAd>)→Z 汎用レジスタまたはメモリのオペランドの指定された1ビットをテストし、ゼロフラグに反映します。ビット番号は、3ビットのイミディエイトデータまたは汎用レジスタの内容下位3ビットで指定されます。
	BAND	B	C∧(<ビット番号>of<EAd>)→C 汎用レジスタまたはメモリのオペランドの指定され1ビットとキャリフラグとの論理積をとり、結果をキャリフラグに格納します。
	BIAND	B	C∧〔～(<ビット番号>of<EAd>)〕→C 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグとの論理積をとり、結果をキャリフラグに格納します。 ビット番号は、3ビットのイミディエイトデータで指定されます。
	BOR	B	C∨(<ビット番号>of<EAd>)→C 汎用レジスタまたはメモリのオペランドの指定され1ビットとキャリフラグとの論理和をとり、結果をキャリフラグに格納します。
	BIOR	B	C∨〔～(<ビット番号>of<EAd>)〕→C 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグとの論理和をとり、結果をキャリフラグに格納します。 ビット番号は、3ビットのイミディエイトデータで指定されます。

【００５１】　　【表１２】

40

50

25　　26

表１２

命令の機能別一覧(5)

分類	命令	サイズ*	機　　能
ビット操作命令	BXOR	B	C⊕(<ビット番号>of<EAd>)→C 汎用レジスタまたはメモリのオペランドの指定された1ビットとキャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。
	BIXOR	B	C⊕〔～(<ビット番号>of<EAd>)〕→C 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグとの排他的論理和をとり、結果をキャリフラグに格納します。 ビット番号は、3ビットのイミディエイトデータで指定されます。
	BLD	B	(<ビット番号>of<EAd>)→C 汎用レジスタまたはメモリのオペランドの指定された1ビットをキャリフラグに転送します。
	BILD	B	～(<ビット番号>of<EAd>)→C 汎用レジスタまたはメモリのオペランドの指定された1ビットを反転し、キャリフラグに転送します。 ビット番号は、3ビットのイミディエイトデータで指定されます。
	BST	B	C→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された1ビットに、キャリフラグの内容を転送します。
	BIST	B	～C→(<ビット番号>of<EAd>) 汎用レジスタまたはメモリのオペランドの指定された1ビットに、キャリフラグを反転して転送します。 ビット番号は、3ビットのイミディエイトデータで指定されます。

【００５２】　　【表１３】

30

40

50

27　　28

表１３

命令の機能別一覧(6)

<table>
<tr><th>分類</th><th>命令</th><th>サイズ*</th><th>機　　能</th></tr>
<tr><td rowspan="8">分岐命令</td><td>Bcc</td><td>—</td><td>指定した条件が成立しているとき、指定されたアドレスへ分岐します。分岐条件を下表に示します。
<table>
<tr><th>ニーモニック</th><th>説明</th><th>分岐条件</th></tr>
<tr><td>BRA(BT)</td><td>Always(True)</td><td>Always</td></tr>
<tr><td>BRN(BF)</td><td>Never(False)</td><td>Never</td></tr>
<tr><td>BHI</td><td>HIgh</td><td>C∨Z=0</td></tr>
<tr><td>BLS</td><td>Low or Same</td><td>C∨Z=1</td></tr>
<tr><td>BCC(BHS)</td><td>Carry Clear(High or Same)</td><td>C=0</td></tr>
<tr><td>BCS(BLO)</td><td>Carry Set(LOw)</td><td>C=1</td></tr>
<tr><td>BNE</td><td>Not Equal</td><td>Z=0</td></tr>
<tr><td>BEQ</td><td>EQual</td><td>Z=1</td></tr>
<tr><td>BVC</td><td>oVerflow Clear</td><td>V=0</td></tr>
<tr><td>BVS</td><td>oVerflow Set</td><td>V=1</td></tr>
<tr><td>BPL</td><td>PLus</td><td>N=0</td></tr>
<tr><td>BMI</td><td>MInus</td><td>N=1</td></tr>
<tr><td>BGE</td><td>Greater or Equal</td><td>N⊕V=0</td></tr>
<tr><td>BLT</td><td>Less Than</td><td>N⊕V=1</td></tr>
<tr><td>BGT</td><td>Greater Than</td><td>Z∨(N⊕V)=0</td></tr>
<tr><td>BLE</td><td>Less or Equal</td><td>Z∨(N⊕V)=1</td></tr>
</table></td></tr>
<tr><td>JMP</td><td>—</td><td>指定されたアドレスへ無条件に分岐します。</td></tr>
<tr><td>BSR</td><td>—</td><td>指定されたアドレスへサブルーチン分岐します。</td></tr>
<tr><td>JSR</td><td>—</td><td>指定されたアドレスへサブルーチン分岐します。</td></tr>
<tr><td>RTS</td><td>—</td><td>サブルーチンから復帰します。</td></tr>
</table>

【００５３】　　【表１４】

30

40

50

表１４

命令の機能別一覧(7)

分類	命令	サイズ*	機　　能
システム制御命令	TRAPA	—	命令トラップ例外処理を行います。
	RTE	—	例外処理ルーチンから復帰します。
	SLEEP	—	低消費電力状態に遷移します。
	LDC	B／W	(EAs)→CCR、(EAs)→EXR 汎用レジスタまたはメモリの内容をCCR、EXRに転送します。また、イミディエイトデータをCCR、EXRに転送します。CCR、EXRは8ビットですが、メモリとCCR、EXR間の転送はワードサイズで行われ、上位8ビットが有効になります。
	STC	B／W	CCR→(EAd)、EXR→(EAd) CCR、EXRの内容を汎用レジスタまたはメモリに転送します。CCR、EXRは8ビットですが、CCR、EXRとメモリ間の転送はワードサイズで行われ、上位8ビットが有効になります。
	ANDC	B	CCR∧#IMM→CCR、EXR∧#IMM→EXR CCR、EXRとイミディエイトデータの論理積をとります。
	ORC	B	CCR∨#IMM→CCR、EXR∨#IMM→EXR CCR、EXRとイミディエイトデータの論理和をとります。
	XORC	B	CCR⊕#IMM→CCR、EXR⊕#IMM→EXR CCR、EXRとイミディエイトデータの排他的論理和をとります。
	NOP	—	PC+2→PC PCのインクリメントだけを行います。

【００５４】

【表１５】

表１５

命令の機能別一覧(8)

分類	命令	サイズ*	機　　能
ブロック転送命令	EEPMOV.B	—	if R4L≠0 then Repeat @ER5+→@ER6+ R4L－1→R4L Until R4L=0 else next;
	EEPMOV.W	—	if R4≠0 then Repeat @ER5+→@ER6+ R4－1→R4 Until R4=0 else next; ブロック転送命令です。ER5で示されるアドレスから始まり、R4LまたはR4で指定されるバイト数のデータを、ER6で示されるアドレスのロケーションへ転送します。転送終了後、次の命令を実行します。

【００５５】基本的な命令は平成５年６月（株）日立製作所発行『Ｈ８／３００Ｈシリーズプログラミングマニュアル』などに記載のＣＰＵと同様であり、いわゆる、ロードストアアーキテクチャを採用している。命令とアドレッシングモードの組み合わせを削減し、ＣＰＵの命令制御の論理規模・物理的規模を縮小できる。

【００５６】本発明のＣＰＵは、上記従来ＣＰＵに対して命令実行時間の高速化を実現している。

【００５７】ＣＰＵの命令は、２バイト（ワード）を単位にしている。各命令は下記のようなオペレーションフィールド（ｏｐ）、レジスタフィールド（ｒ）、ＥＡ拡張部（ＥＡ）、およびコンディションフィールド（ｃｃ）から構成されている。

【００５８】（１）オペレーションフィールド
命令の機能を表し、アドレッシングモードの指定、オペランドの処理内容を指定する。命令の先頭４ビットを必ず含んでいる。２つのオペレーションフィールドを持つ場合もある。

【００５９】（２）レジスタフィールド
汎用レジスタを指定する。アドレスレジスタのとき３ビット、データレジスタのとき３ビットまたは４ビットである。２つのレジスタフィールドを持つ場合、またはレジスタフィールドを持たない場合もある。

【００６０】（３）ＥＡ拡張部
イミディエイトデータ、絶対アドレスまたはディスプレースメントを指定する。８ビット、１６ビット、または３２ビットである。

【００６１】（４）コンディションフィールド
Ｂｃｃ命令の分岐条件を示す。

【００６２】図３に、命令の基本フォーマットの例を示す。

【００６３】図４に、マイクロコンピュータにおいて、ＣＰＵ１に対し乗算器２を取外し可能に設けた概略ブロック図を示す。命令レジスタ（ＩＲ）２１、命令デコーダ・制御回路（ＣＯＮＴ）２２、レジスタセレクタ（ＲＳＥＬ）２３、ライトデータバッファ（ＤＢＷ）２４、リードデータバッファ（ＤＢＲ）２５、演算器（ＡＬＵ）２６、演算器（ＩＮＣ）２７、汎用レジスタ（ＥＲ０～ＥＲ７）２８Ａ～２８Ｈ、エミュレータスタックポインタ（ＥＭＬＳＰ）２９、プログラムカウンタ（ＰＣ）３０、コンディションコードレジスタ（ＣＣＲ）３１、拡張レジスタ（ＥＸＲ）３２、アドレスバッファ（ＭＡＢ）３３からなる。乗算器２なしのＣＰＵ１はこれらによって構成される。各バッファやレジスタ、演算器の各ブロックの機能は、特開平５－２４１８２６号公報に記載のＣＰＵと概略同様である。また、乗算器２を含むＣＰＵ１は、更に、バススイッチ３４、乗算器２がある。

【００６４】命令デコーダ・制御回路（ＣＯＮＴ）２２には、制御信号ＣＰＵＳ、制御信号ＩＮＴＭ１、そのほかの制御信号（割り込み要求など）が入力されている。ＣＯＮＴ２２は各部を制御するための、出力タイミングの相違する制御信号Ａ、Ｂ、Ｃを出力する。

【００６５】なお、図中のＣ１およびＣ２は、当該信号の同期タイミングを示す。例えば、ＲＳＥＬ入力１のＣ１はφに同期して入力が行われることを示し、ＲＳＥＬ入力２のＣ２はφ＃（＃は論理反転）に同期して入力が行われることを示す。また、ＡＬＵ入力のＣ１は、φの期間に入力が行われることを示し、ＡＬＵ出力のＣ２は、φ＃の期間に出力が行われることを示す。ＡＬＵ２６とＩＮＣ２７は、それぞれ動作タイミングの異なった演算器であり、それぞれ、オーバラップしつつ演算可能である。

【００６６】そのほかのレジスタなどは、φ、φ＃の両方でデータを入出力可能である。ＧＢ、ＤＢ、ＷＢの各バスはφ、φ＃の両方で異なったデータを転送可能である。

【００６７】φ、φ＃は互いにノーオーバラップの関係の２相クロックとしてもよい。

【００６８】レジスタセレクタ（ＲＳＥＬ）２３には、ＩＲ２１乃至ＣＯＮＴ２２から命令コードの一部（レジスタ指定フィールド）が与えられる。この供給タイミングは、レジスタ指定フィールドの位置によって相違される。ＲＳＥＬ２３は出力タイミングの相違するレジスタ選択信号Ａ、Ｂを出力する。

【００６９】例えば、平成５年６月（株）日立製作所発行『Ｈ８／３００Ｈシリーズプログラミングマニュアル』に記載のＣＰＵにおいては、１６ビット単位の命令コードのビット７－４が、ＣＯＮＴ２２と同時に与えられ（ＲＳＥＬ入力１）、ビット１１－８および３－０（ＲＳＥＬ入力２）が、ＣＯＮＴ２２の内容と０．５ステート遅れて与えられる。ＲＳＥＬ入力２の反転制御信号をＲＳＥＬに与える。

【００７０】ＣＰＵ１内部のＤＢＷ２４、ＤＢＲ２５、ＡＬＵ２６、ＩＮＣ２７、ＥＲ０～ＥＲ７（２８Ａ～２８Ｈ）、ＰＣ３０、ＣＣＲ３１、ＶＡＧ、ＡＢは、ＧＢバス、ＤＢバス、ＷＢバスによって相互に接続されている。

【００７１】２つの演算器ＡＬＵ２６、ＩＮＣ２７に対し、ＧＢ、ＤＢバスからデータを入力し、ＷＢバスにデータを出力する。それぞれの入出力バスの数に対応した数の内部バスとして、バス即ち配線の増加による物理的規模の増加を抑止している。

【００７２】また、ライトデータバッファ（ＤＢＷ）２４は内部データバスへの出力、リードデータバッファ（ＤＢＲ）２５は内部データバスからの入力、アドレスバッファ（ＭＡＢ）３３は内部アドレスバスへの出力、命令レジスタは内部データバスからの入力が可能であり、それぞれ内部バスに接続されている。ライトデータバッファ（ＤＢＷ）２４およびリードデータバッファ（ＤＢＲ）２５は３２ビット構成とされる。ライトデータは３２ビット一括してライトデータバッファ（ＤＢＷ）２４に書き込むことができ、所定のタイミングで、１６ビットの内部データバスに出力される。また、内部データバスから読み出したデータを、リードデータバッファ（ＤＢＲ）２５に一旦格納して、３２ビットのリードデータを一括して出力することができる。ＭＡＢは＋２のインクリメント機能を有する。

【００７３】命令デコーダ・制御回路（ＣＯＮＴ）２２が、ＩＲ２１からの入力、ＣＰＵＳ信号、ＩＮＴＭ１信号やそのほかの入力信号に基づいて、動作制御を行なう。制御回路の出力は所定のバッファを介して出力される。ＣＯＮＴ２２自身にも、ステート番号などがフィードバックされる。

【００７４】アドレスバッファ（ＭＡＢ）３３はインクリメント機能（＋２）を有する。ＥＲ０～ＥＲ７（２８Ａ～２８Ｈは）データレジスタまたはアドレスレジスタとして使用することができる。

【００７５】ＥＭＬＳＰ２９は、ユーザには公開されていないリソースで、エミュレータに搭載されて動作するとき、ユーザプログラムとエミュレーションプログラムの間の遷移時のスタックポインタとして使用する。その内容を指定するために、一部の内容が、ＣＰＵ外部から与えられる。

【００７６】ＰＣ３０は３２ビットのカウンタであり、ＣＰＵ１が次に実行する命令のアドレスを示している。コンデションコードレジスタ（ＣＣＲ）３１は割り込みマスクビット（Ｉ）、キャリフラグ（Ｃ）、ゼロフラグ（Ｚ）、ネガティブフラグ（Ｎ）、オーバフローフラグ（Ｖ）を含んでいる。

【００７７】ＣＰＵ１と乗算器２は、バススイッチ３４を介して接続されている。また、バススイッチ３４は内部データバスとのインタフェースも行う。また、ＣＰＵ１から乗算器２への制御信号を与える。乗算器２のステータス信号ＢＵＳＹと、フラグ検出信号をＣＰＵ１に与える。ＴＥＳＴＭＯＤＥ信号を、例えば、ＳＹＳＣ３から与える。制御信号ＣＰＵＳは、ＳＹＳＣＲ１４あるいはそのほかのレジスタの制御ビットの出力にしてもよいし、マイクロコンピュータの制御端子のようなもので指定してもよい。

【００７８】図５に、制御信号ＣＰＵＳを制御レジスタ（ＣＰＵＣＲ）１４の制御ビットで構成した具体的な例を示す。図は１ビットの構成を示している。ＣＰＵＣＲ１４は、フリップフロップで構成される。フリップフロップにはリセット信号が与えられる。フリップフロップのクロックは内部ライト信号と、アドレスをデコードして得られるＣＰＵＣＲ選択信号の論理積信号とされる。データ入力はデータバスのビット８とされる。出力がＣＰＵＳ信号とされる。また、クロックトバッファＣＢＦ６を介して、データバスに出力される。クロックトバッファＣＢＦ６のクロックは内部ライト信号とＣＰＵＣＲ選択信号の論理積信号とされる。

【００７９】本レジスタのライトは、テストモードや、エミュレータに搭載した場合のブレークモードなどでのみライト可能にするとよい。ブレークモードなどについては、特開平６－１５００２６などに記載されている。同様に、ＴＥＳＴＭＯＤＥ信号を生成することができる。同一のレジスタに配置することができる。

【００８０】図６に、制御信号ＣＰＵＳの設定方法の一例として、エミュレーション用プロセッサおよびエミュレータをブロック図で示す。エミュレーション用プロセッサ３８は、マイクロコンピュータ部分にエミュレーション用インタフェース３９を加えて構成される。エミュレーション用インタフェース３９には、エミュレーション用プロセッサ専用の制御レジスタ４１を有する。メモリ４２は、ＲＯＭ、ＲＡＭを含み、Ｉ／ＯはＩ／Ｏポート、タイマ、ＳＣＩなどを含む。

【００８１】コネクタ部がマイクロコンピュータの代わりに応用システム（ユーザシステム）４３に装着される。エミュレーション用プロセッサ３８は上記コネクタ部とインタフェースケーブル４４を介し、ターゲットシステムインタフェースを用いて上記応用システム４３と信号の入出力を行う。

【００８２】応用システム（ユーザシステム）４３には、特に制限はされないものの、ユーザバス４５が存在し、ユーザメモリ４６が接続される。エミュレーション用プロセッサ３８が出力し、インタフェースケーブル４４を介して供給されるユーザストローブ信号に従って、ユーザメモリ４６はリード／ライトされる。

【００８３】一方、エミュレーション用プロセッサ３８は上記エミュレーションインタフェース３９を用いてエミュレーションバス４７に接続される。エミュレーションバス４７には図示はされない状態信号・制御信号などを含む。上記エミュレーションバス４７を用いて、エミュレーション用プロセッサ３８から、応用システム４３とエミュレーション用プロセッサ３８の内部状態に応じた情報などが出力され、また、エミュレーション用プロセッサ３８に対し、エミュレーションのための各種制御信号が入力される。エミュレーション用プロセッサ３８の、図示はされないエミュレートモード端子が電源レベルに固定され、エミュレーション用プロセッサ３８内部ではエミュレートモードが設定される。

【００８４】さらに、上記エミュレーションバス４７には、特に制限はされないものの、応用システム４３またはターゲットマイクロコンピュータ内蔵のメモリを代行するためのＲＡＭでなるようなエミュレーションメモリ４８がある。また、エミュレーション用プロセッサ３８の制御状態やエミュレーションバス４７の状態を監視して、その状態が予め設定された状態に達した時に、上記エミュレータ専用割込みを入力して、ＣＰＵによるユーザプログラムの実行を停止させ、エミュレーション用プログラム実行状態に遷移させる（ブレーク）ためのブレーク制御回路４９と、上記ＣＰＵのリード動作またはライト動作を示す信号、命令リード動作を示す信号などに基づき、エミュレーションバス４７に与えられるアドレスデータさらには制御情報を逐次蓄えるリアルタイムトレース回路５０などが接続される。

【００８５】上記エミュレーションバス４７が、エミュ

レーションメモリ４８、ブレーク制御回路４９、リアルタイムトレース回路５０などに、それぞれ接続される。これらでもってマイクロコンピュータ開発装置５５が構成されている。

【００８６】上記エミュレーションメモリ４８、ブレーク制御回路４９、リアルタイムトレース回路５０はコントロールバス５１に接続され、コントロールバス５１を介してコントロールプロセッサ５２の制御を受けるようになっている。上記コントロールバス５１は、エミュレーション用プロセッサ制御回路に接続されるとともに、インタフェース回路を介して、特に制限はされないもののパーソナルコンピュータなどのシステム開発装置５４に接続される。例えば、システム開発装置５４から入力されたプログラムをエミュレーションメモリ４８に転送し、内蔵ＲＯＭ上に配置されるべきかかるプログラムをＣＰＵ１がリードすると、エミュレーションメモリ４８上のプログラムがリードされる。また、ブレーク条件や、リアルタイムトレース条件などもシステム開発装置５４から与えることができる。

【００８７】コントロールプロセッサ５２は、ＣＰＵＳ信号をエミュレーション用プロセッサ３８に供給して、乗算器の使用／不使用の選択を行うことができる。コントロールプロセッサ５２は、システム開発装置５４から入力された情報などに基づいて、ＣＰＵＳ信号を制御する。あるいは、図５のような制御レジスタを、エミュレーション用インタフェース３９内に制御レジスタに設けて、エミュレータ４０のソフトウェアをＣＰＵが実行して、前記制御レジスタを指定することによって、ＣＰＵＳ信号を生成するようにすることができる。この場合は、エミュレーション用ソフトウェアの実行モード、いわゆるブレークモードでのみライト可能にすると都合がよい。開発途上にあるユーザのソフトウェアの誤動作によって、誤った設定を行うことがない。

【００８８】エミュレーション用プロセッサ３８およびエミュレータ４０を複数のＣＰＵをサポート可能にすることによって、実際のマイクロコンピュータのみを開発すればよく、開発効率を向上することができる。なお、ＥＭＬＳＰ２９のアドレス指定情報も、エミュレーション用インタフェース３９内の制御レジスタで指定することができる。

【００８９】エミュレーション用プロセッサ３８やエミュレータ４０については、特開平３－２７１８３４号公報、あるいは特開平６－１５００２６号公報などに記載されている。

【００９０】図７に、制御信号ＣＰＵＳ設定方法の一例である、マイクロコンピュータの主要部をブロック図で示す。ＣＰＵＳ信号をレジスタによらず、ＣＭＯＳインバータ回路５８の出力とする。かかるＣＭＯＳインバータ回路５８は、Ｐチャネル型ＭＯＳトランジスタＱ１、Ｎチャネル型ＭＯＳトランジスタＱ２で構成される。このＣＭＯＳインバータ回路５８の入力は、抵抗Ｒを介して電源Ｖｄｄに接続されると共に、保護回路Ｑ３、Ｑ４を介して端子Ｐに結合される。端子Ｐは、ワイヤＷによってグランドレベル電源用リードＬに接続されるか、解放状態とされるかが選択され、ＣＰＵＳの設定を行なう。

【００９１】端子Ｐが解放状態とされれば、ＣＭＯＳインバータ回路５８の入力はハイレベルとなって、ＣＰＵＳ信号は非活性状態になる。一方、端子Ｐが、ワイヤＷによって、グランドレベル電源用リードＬに接続されれば、ＣＭＯＳインバータ回路５８の入力はロウレベルとなって、ＣＰＵＳ信号は活性状態になる。乗算器を使用可能にする。

【００９２】端子Ｐは対応するリードを持たず、例えばプラスティックパッケージに封止された場合には、対応する端子を持たない。

【００９３】これにより、半導体集積回路装置のパッケージの端子を直接利用することなく、乗算器の制御を設定できるため、一定のパッケージを用いた場合に、有効な端子数の減少防ぐことができる。この場合、端子Ｐをグランドレベル電源端子に隣接して配置すると都合がよい。

【００９４】あるいは、端子Ｐをグランドレベル電源用リードＬにワイヤＷによって接続するか、しないかの選択を、半導体集積回路装置の配線変更として実現してもよい。ＣＭＯＳインバータ回路５８の入力を、半導体集積回路装置内部の電源電圧またはグランドのいずれに接続するかを選択すればよい。このとき、抵抗Ｒ及び端子Ｐは削除することができる。または、ＣＰＵＳビットをＰＲＯＭ素子などで構成してもよい。この場合、製造者が設定を行なってもよいし、ユーザが設定を行なってもよい。

【００９５】図８および図９に、ＣＰＵの内部レジスタ構成を示す。これらのレジスタは、図８の汎用レジスタおよび図９のコントロールレジスタの２つに分割される。以下、各レジスタについて説明する。

【００９６】（１）汎用レジスタ

ＣＰＵはこの汎用レジスタを８本有している。この汎用レジスタは３２ビット長からなり、すべて同じ機能を有しており、アドレスレジスタとしてもデータレジスタとしても使用することができる。データレジスタとしては３２ビット、１６ビットおよび８ビットレジスタとして使用できる。

【００９７】アドレスレジスタ及び３２ビットレジスタとしては、一括して汎用レジスタＥＲ（ＥＲ０～ＥＲ７）として使用する。１６ビットレジスタとしては、汎用レジスタＥＲを分割して汎用レジスタＥ（Ｅ０～Ｅ７）、汎用レジスタＲ（Ｒ０～Ｒ７）として使用する。これらは同等の機能を有しており、１６ビットレジスタを最大１６本まで使用することができる。

【００９８】８ビットレジスタとしては、汎用レジスタＲを分割して汎用レジスタＲＨ（Ｒ０Ｈ～Ｒ７Ｈ）、汎用レジスタＲＬ（Ｒ０Ｌ～Ｒ７Ｌ）として使用する。これらは同等の機能を有しており、８ビットレジスタを最大１６本まで使用することができる。

【００９９】図１０に、汎用レジスタの使用方法を示す。各レジスタは独立して使用方法を選択することができる。

【０１００】汎用レジスタＥＲ７には、汎用レジスタとしての機能に加えて、スタックポインタ（ＳＰ）としての機能が割り当てられており、例外処理やサブルーチン分岐などで暗黙的に使用される。図１１にスタックの状態を示す。

【０１０１】（２）コントロールレジスタ

コントロールレジスタは、２４ビットのプログラムカウンタ（ＰＣ）と８ビットの拡張レジスタ（エクステンドレジスタ）（ＥＸＲ）および８ビットのコンディションコードレジスタ（ＣＣＲ）を含んでいる。

【０１０２】①プログラムカウンタ（ＰＣ）

２４ビットのカウンタで、ＣＰＵが次に実行する命令のアドレスを示している。ＣＰＵの命令は、すべて２バイト（ワード）を単位としているため、最下位ビットは無効である。（命令コードのリード時には最下位ビットは”０”とみなされる）。

【０１０３】分岐命令の実行アドレスの上位８ビットは無視される。プログラム領域として使用できるのは、Ｈ’００００００００～Ｈ’００ＦＦＦＦＦＦの領域である。

【０１０４】②拡張レジスタ（ＥＸＲ）

８ビットのレジスタで、トレースビット（Ｔ）、割込みマスクビット（Ｉ２～Ｉ０）を含む８ビットで構成されている。

【０１０５】ビット７：トレースビット（Ｔ）

トレースビットか否かを指定する。本ビットが”０”にクリアされているときは命令を順次実行する。”１”にセットされているときは１命令実行する毎にトレース例外処理を実行する。

【０１０６】ビット６～４：リザーブビット

リザーブビットである。

【０１０７】ビット２～０：割込みマスクビット（Ｉ２～Ｉ０）

割込み要求マスクレベル（０～７）を指定する。

【０１０８】ＥＸＲは、ＬＤＣ、ＳＴＣ、ＡＮＤＣ、ＯＲＣ、ＸＯＲＣ命令で実行することができる。このうち、ＳＴＣを除く命令を実行した場合、実行終了後３ステートの間は、ＮＭＩを含めてすべての割込みは受け付けられない。

【０１０９】③コンディションコードレジスタ（ＣＣＲ）

８ビットのレジスタで、ＣＰＵの内部状態を示す。割込みマスクビット（Ｉ）とハーフキャリ（Ｈ）、ネガティブ（Ｎ）、ゼロ（Ｚ）、オーバフロー（Ｖ）、キャリ（Ｃ）を含む８ビットで構成されている。

【０１１０】ビット７：割込みマスクビット（Ｉ）

本ビットが”１”にセットされると、割込みがマスクされる。ただし、ＮＭＩはＩビットに関係なく受け付けられる。例外処理の実行が開始されたときに”１”にセットされる。

【０１１１】ビット６：ユーザビット／割込みマスクビット（ＵＩ）

ソフトウエア（ＬＤＣ、ＳＴＣ、ＡＮＤＣ、ＯＲＣ、ＺＯＲＣ命令）でリード／ライトできる。割込みマスクビットとしても使用可能である。

【０１１２】ビット５：ハーフキャリフラグ（Ｈ）

ＡＤＤ．Ｂ、ＡＤＤＸ．Ｂ、ＳＵＢ．Ｂ、ＳＵＢＸ．Ｂ、ＣＭＰ．Ｂ、ＮＥＧ．Ｂ命令の実行により、ビット３にキャリまたはボローが生じたとき”１”にセットされ、生じなかったとき”０”にクリアされる。また、ＡＤＤ．Ｗ、ＳＵＢ．Ｗ、ＣＭＰ．Ｗ、ＮＥＧ．Ｗ命令の実行により、ビット１１にキャリまたはボローが生じたとき、ＡＤＤ．Ｌ、ＳＵＢ．Ｌ、ＣＭＰ．Ｌ、ＮＥＧ．Ｌ命令の実行により、ビット２７にキャリまたはボローが生じたとき、”１”にセットされ、生じなかったとき”０”にクリアされる。

【０１１３】ビット４：ユーザビット（Ｕ）

ソフトウエア（ＬＤＣ、ＳＴＣ、ＡＮＤＣ、ＯＲＣ、ＸＯＲＣ命令）でリート／ライトできる。

【０１１４】ビット３：ネガティブフラグ（Ｎ）

データの最上位ビットを符号ビットとみなし、最上位ビットの値を格納する。

【０１１５】ビット２：ゼロフラグ（Ｚ）

データがゼロのとき”１”にセットされ、ゼロ以外のとき”０”にクリアされる。

【０１１６】ビット１：オーバフローフラグ（Ｖ）

算術演算命令により、オーバフローが生じたとき”１”にセットされる。それ以外のとき”０”にクリアされる。

【０１１７】ビット０：キャリフラグ（Ｃ）

演算の実行により、キャリが生じたとき”１”にセットされ、生じなかったとき”０”にクリアされる。キャリには次の種類がある。

【０１１８】（ａ）加算結果のキャリ

（ｂ）減算結果のボロー

（ｃ）シフト／ローテートのキャリ

また、キャリフラグには、ビットアキュムレータの機能があり、ビット操作命令で使用される。なお、命令によってはフラグが変化しない場合がある。ＣＣＲは、ＬＤＣ、ＳＴＣ、ＡＮＤＣ、ＯＲＣ、ＸＯＲＣ命令で操作することができる。また、Ｎ、Ｚ、Ｖ、Ｃの各フラグは、条件分岐命令（Ｂｃｃ）で使用される。

【０１１９】④積和レジスタ（ＭＡＣ）
６４ビットのレジスタであり、積和演算結果を格納する。３２ビットのＭＡＣＨ、ＭＡＣＬから構成される。ＭＡＣＨは下位１０ビットが有効であり、上位は符号拡張されている。
【０１２０】図１２に、ＣＰＵの基本動作タイミングを示す。
【０１２１】ＡＤＤ．Ｗ　Ｒ０、Ｒ１のようなレジスタ間演算のタイミングである。特に制限はされないものの、内部データバスは１６ビットであって、内蔵ＲＯＭ、ＲＡＭのリード／ライトを１ステートでリード／ライト可能とする。
【０１２２】Ｔ０のＣ２（φ＃同期。＃は反転論理を示す）で、ＣＰＵ１のアドレスバッファ（ＭＡＢ）３３からアドレスがＩＡＢに出力される。
【０１２３】Ｔ１のＣ１（φ同期）で、ＩＡＢの内容がＰＡＢに出力され、リードサイクルが開始される。Ｃ２でリードデータが内部データバスに得られ、これをＩＲ２１にラッチする。以上の動作は以前の命令の実行の制御によって行われる。
【０１２４】直前の命令の実行が終了すると、最も早く命令の実行が開始される場合には、Ｔ２のＣ１で命令コードがＣＯＮＴ２２に入力されて、命令の内容が解読される。解読結果に従って、制御信号を出力して、各部の制御を行う。命令の一部（レジスタ指定フィールド：ＲＳＥＬ入力信号１）がレジスタセレクタ２３に与えられる。
【０１２５】レジスタ間演算命令では、Ｔ２のＣ２で、ＰＣの内容を内部バスＧＢに読み出して、ＭＡＢ３３とＩＮＣ２７に入力する。ＭＡＢ３３からアドレスＩＡＢが出力される。レジスタセレクタ２３に制御信号を与える。ＲＳＥＬ入力信号１と制御信号Ａ（Ｒｓ－ＤＢ出力、Ｒｄ－ＧＢ出力）とに基づいて、レジスタ選択信号Ｂが生成される。ＲＳＥＬ入力信号２がレジスタセレクタ２３に与えられる。
【０１２６】Ｔ３から、次の次の命令がリードされる。Ｔ３のＣ１で、ＩＮＣ２７でインクリメント（＋２）された結果が、内部バスＷＢを経由して、ＰＣ３０にライトされる。ＲＳＥＬ入力信号２と制御信号Ｂ（ＷＢ－Ｒｄ入力）とに基づいて、レジスタ選択信号Ｃが生成される。レジスタ選択信号Ｂがレジスタを選択して、ソース側、デスティネーション側のレジスタ（Ｓ、Ｄ）のデータをＡＬＵ２６に入力する。ＡＬＵ２６の演算内容はＣＯＮＴ２２が制御信号Ｃによって指示する。加減算・論理演算・シフトなどは１クロックで演算を行うことができる。例えば、上記命令では１６ビットの加算を行う。次の命令のＣＯＮＴ２２へのロードを指示する。ＲＳＥＬ入力信号２と制御信号Ｂ（ＷＢ－Ｒｄ入力）とに基づいて、レジスタ選択信号Ｃが生成される。
【０１２７】Ｔ３のＣ２で、ＡＬＵ２６の演算結果（Ｒ）が、内部バスＷＢを経由して、レジスタ選択信号Ｃが選択したデスティネーション側のレジスタにライトされる。制御信号Ｃによって、ＣＣＲ３１の更新を行う。更に次の次の命令をＩＲ２１に取り込む。同時に、次の命令の実行が開始され、例えば、ＰＣ３０の内容を読み出して、ＭＡＢ３３とＩＮＣ２７に入力される。レジスタ間演算を実質的に１ステートで実行できる。２つの演算器２６、２７の入出力バスの数に対応した数の内部バスとして（演算器に対応して、内部バスを増加させることなく）、バス即ち配線の増加による物理的規模の増加を抑止している。
【０１２８】図１３に、ＣＰＵの基本動作タイミングを示す。
【０１２９】ＭＯＶ．Ｗ　Ｒ０、@Ｒ１のような、レジスタ間接によるデータライトのタイミングである。
【０１３０】Ｔ０のＣ２で、ＣＰＵ１のＭＡＢ３３からアドレスがＩＡＢに出力される。
【０１３１】Ｔ１のＣ１で、アドレスがＰＡＢに出力され、リードサイクルが開始される。Ｃ２でリードデータが内部データバスに得られ、これをＩＲ２１にラッチする。
【０１３２】直前の命令の実行が終了すると、Ｔ２のＣ１で命令コードがＣＯＮＴ２２に入力されて、命令の内容が解読され、各部の制御を行う。命令の一部のレジスタ指定フィールド（ＲＳＥＬ入力信号１）がレジスタセレクタ２３に与えられる。レジスタ間接によるデータライトでは、制御信号ＡとＲＳＥＬ入力信号１とに基づいて、レジスタ選択信号Ａが与えられ、アドレスとして指定されたレジスタが選択される。
【０１３３】Ｔ２のＣ２で、選択されたレジスタの内容（Ａ）を内部バスＧＢに読み出して、ＭＡＢ３３を経由してアドレスＩＡＢに出力される。ＲＳＥＬ入力信号２がレジスタセレクタ２３に与えられる。ＲＳＥＬ入力信号２と制御信号Ｂ（Ｒｄ－ＤＢ出力）とに基づいて、レジスタ選択信号Ｂが生成される。
【０１３４】Ｔ３のＣ１で、制御信号Ｃの一部がＣＯＮＴ２２に入力され、状態遷移が行われる（ステートマシンが構成される）。ＩＡＢの内容に基づいて、ライトサイクルが開始される。選択されたレジスタの内容（Ｄ）を内部バスＤＢに読み出して、データバッファ（ＤＢＷ）を経由して内部データバスに出力される。
【０１３５】Ｔ３のＣ２で、ＰＣ３０の内容を内部バスＧＢに読み出して、ＭＡＢ３３とＩＮＣ２７に入力する。ＭＡＢ３３からアドレスＩＡＢが出力される。ＲＳＥＬ入力信号１と制御信号Ａ（Ｒｄ－ＧＢ出力）とに基づいて、レジスタ選択信号Ｂが生成される。
【０１３６】Ｔ４から、次の次の命令がリードされる。
【０１３７】Ｔ４のＣ１で、ＩＮＣ２７でインクリメント（＋２）された結果が、内部バスＷＢを経由して、ＰＣ３０にライトされる。レジスタ選択信号Ｂがレジスタ

を選択して、データレジスタ（D）のデータをＡＬＵ２６に入力する。ＡＬＵ２６の演算内容はＣＯＮＴ２２が制御信号Ｃによって指示する。転送の場合はデータのチェックのみを行う。次の命令のＣＯＮＴ２２へのロードを指示する。

【０１３８】Ｔ４のＣ２で、制御信号Ｃによって、チェックした結果によって、ＣＣＲ３１の更新を行う。更に次の次の命令をＩＲ２１に取り込む。同時に、次の命令の実行が開始され、例えば、ＰＣ３０の内容を読み出して、ＭＡＢ３３とＩＮＣ２７に入力される。

【０１３９】ＲＳＥＬに入力するタイミングを、ＲＳＥＬ入力１（アドレスレジスタ、ソースレジスタ）とＲＳＥＬ入力２（データレジスタ、ディスティネーションレジスタ）のように、レジスタ指定フィールド毎にことなったタイミング（φ同期とφ＃同期）とすることにより、命令実行の高速化を実現することができる。

【０１４０】表１６乃至表１９に、本発明に関係のある命令の説明を示す。

【０１４１】表１６は命令コードを示し、表１８は命令の実行状態を示し、表１９はコンディションコードの変化を示している。表１７はレジスタフィールドと汎用レジスタの対応を示している。

【０１４２】

【表１６】

表16

命令コード

命令	ニーモニック	サイズ	インストラクションフォーマット 第1バイト	第2バイト	第3バイト	第4バイト	第5バイト	第6バイト	第8バイト	第10バイト
CLRMAC	CLRMAC	-	0 1	A 0						
DIVXS	DIVXS.B Rs,Rd	B	0 1	D 0	5 1	rs rd				
	DIVXS.W Rs,ERd	W	0 1	D 0	5 3	rs 0 erd				
DIVXU	DIVXU.B Rs,Rd	B	5 1	rs rd						
	DIVXU.W Rs,ERd	W	5 3	rs 0 erd						
LDM	LDM @SP+, (ERn-ERn+1)	L	0 1	1 0	6 D	7 0 ern+1				
	LDM @SP+, (ERn-ERn+2)	L	0 1	2 0	6 D	7 0 ern+2				
	LDM @SP+, (ERn-ERn+3)	L	0 1	3 0	6 D	7 0 ern+3				
LDMAC	LDMAC ERs,MACH	L	0 3	2 0 ers						
	LDMAC ERs,MACL	L	0 3	3 0 ers						
MAC	MAC @ERn+,@ERm+	-	0 1	6 0	6 D	0 ern 0 erm				
MULXS	MULXS.B Rs,Rd	B	0 1	C 0	5 0	rs rd				
	MULXS.W Rs,ERd	W	0 1	C 0	5 2	rs 0 erd				
MULXU	MULXU.B Rs,Rd	B	5 0	rs rd						
	MULXU.W Rs,ERd	W	5 2	rs 0 erd						
POP	POP.W Rn	W	6 D	7 rn						
	POP.L ERn	L	0 1	0 0	6 D	7 0 ern				
PUSH	PUSH.W Rn	W	6 D	F rn						
	PUSH.L ERn	L	0 1	0 0	6 D	F 0 ern				
STM	STM (ERn-ERn+1), @-SP	L	0 1	1 0	6 D	F 0 ern				
	STM (ERn-ERn+2), @-SP	L	0 1	2 0	6 D	F 0 ern				
	STM (ERn-ERn+3), @-SP	L	0 1	3 0	6 D	F 0 ern				
STMAC	STMAC MACH,ERd	L	0 2	2 0 ers						
	STMAC MACL,ERd	L	0 2	3 0 ers						

【表17】

<<記号説明>>

rn, rd, rm: レジスタフィールド（4ビットで8ビットレジスタまたは16ビットレジスタを指定します。
rs、rd、rmはそれぞれオペランド形式のRs、Rd、Rnに対応します。）

ers,erd,ern,ern レジスタフィールド（3ビットでアドレスレジスタまたは32ビットレジスタを指定します。
ers、erd、ern、ermはそれぞれオペランド形式のERs、ERd、ERn、ERmに対応します。）

45　　46

表１７

アドレスレジスタ 32ビットレジスタ		16ビットレジスタ		8ビットレジスタ	
ﾚｼﾞｽﾀﾌｨｰﾙﾄﾞ	汎用ﾚｼﾞｽﾀ	ﾚｼﾞｽﾀﾌｨｰﾙﾄﾞ	汎用ﾚｼﾞｽﾀ	ﾚｼﾞｽﾀﾌｨｰﾙﾄﾞ	汎用ﾚｼﾞｽﾀ
000	ER0	0000	R0	0000	R0H
001	ER1	0001	R1	0001	R1H
：	：	：	：	：	：
：	：	：	：	：	：
：	：	：	：	：	：
111	ER7	0111	R7	0111	R7H
		1000	E0	1000	R0L
		1001	E1	1001	R1L
		：	：	：	：
		：	：	：	：
		：	：	：	：
		1111	E7	1111	R7L

MOV.L ERs,@(d:32,ERd)命令の第４バイト、ビット７は"1","0"のどちらでも動作可能です。

【０１４４】

【表１８】

20

30

40

50

47　　48

表１８

命令の実行状態

命令	1	2	3	4	5	6	7	8	9
CLRMAC	R:W NEXT	内部動作ｽﾀｰﾄ							
DIVXS.B Rs,Rd	R:W 2nd	R:W NEXT	内部動作１１ステート						
DIVXS.W Rs,ERd	R:W 2nd	R:W NEXT	内部動作１９ステート						
DIVXU.B Rs,Rd	R:W NEXT	内部動作１１ステート							
DIVXU.W Rs,ERd	R:W NEXT	内部動作１９ステート							
LDM @SP+,(ERa-ERa+1)	R:W 2nd	R:W NEXT	内部動作ｽﾀｰﾄ	R:W ｽﾀｯｸ(H) *3	R:W ｽﾀｯｸ(L) *3				
LDM @SP+,(ERa-ERa+2)	R:W 2nd	R:W NEXT	内部動作ｽﾀｰﾄ	R:W ｽﾀｯｸ(H) *3	R:W ｽﾀｯｸ(L) *3				
LDM @SP+,(ERa-ERa+3)	R:W 2nd	R:W NEXT	内部動作ｽﾀｰﾄ	R:W ｽﾀｯｸ(H) *3	R:W ｽﾀｯｸ(L) *3				
LDMAC ERs,MACH	R:W NEXT	内部動作ｽﾀｰﾄ		― n回繰り返す *3 →					
LDMAC ERs,MACL	R:W NEXT	内部動作ｽﾀｰﾄ							
MAC @ERa+,@ERa+	R:W 2nd	R:W NEXT	R:W EAn	R:W EAm					
MULXS.B Rs,Rd	R:W 2nd	R:W NEXT	内部動作２ステート						
MULXS.W Rs,ERd	R:W 2nd	R:W NEXT	内部動作３ステート						
MULXU.B Rs,Rd	R:W NEXT	内部動作２ステート							
MULXU.W Rs,ERd	R:W NEXT	内部動作３ステート							
STM (ERa-ERa+1),@-SP	R:W 2nd	R:W NEXT	内部動作ｽﾀｰﾄ	W:W ｽﾀｯｸ(H) *3	W:W ｽﾀｯｸ(L) *3				
STM (ERa-ERa+2),@-SP	R:W 2nd	R:W NEXT	内部動作ｽﾀｰﾄ	W:W ｽﾀｯｸ(H) *3	W:W ｽﾀｯｸ(L) *3				
STM (ERa-ERa+3),@-SP	R:W 2nd	R:W NEXT	内部動作ｽﾀｰﾄ	W:W ｽﾀｯｸ(H) *3	W:W ｽﾀｯｸ(L) *3				
STMAC MACH,ERd	R:W NEXT			― n回繰り返す *3 →					
STMAC MACL,ERd	R:W NEXT								

*3　2本退避／復帰時は2回、3本退避／復帰時は3回、4本退避／復帰時は4回繰り返します。

【０１４５】

40　【表１９】

表１９

コンディションコードの変化

命　令	H	N	Z	V	C	定　　義
CLRMAC	－	－	－	－	－	
DIVXS	－	⇔	⇔	－	－	N=Sm・$\overline{Dm}$＋$\overline{Sm}$・Dm Z=$\overline{Sm}$・$\overline{Sm\text{-}1}$・　……　・$\overline{S0}$
DIVXU	－	⇔	⇔	－	－	N=Sm Z=$\overline{Sm}$・$\overline{Sm\text{-}1}$・　……　・$\overline{S0}$
LDM	－	－	－	－	－	
LDMAC	－	－	－	－	－	
MAC	－	－	－	－	－	
MULXS	－	⇔	⇔	－	－	N=R2m Z=$\overline{R2m}$・$\overline{R2m\text{-}1}$・　……　・$\overline{R0}$
MULXU	－	－	－	－	－	
POP	－	⇔	⇔	0	－	N=Rm Z=$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$
PUSH	－	⇔	⇔	0	－	N=Rm Z=$\overline{Rm}$・$\overline{Rm\text{-}1}$・　……　・$\overline{R0}$
STM	－	－	－	－	－	
STMAC	－	⇔	⇔	⇔	－	N＝MAC命令の結果、MACレジスタが負のとき Z＝MAC命令の結果、MACレジスタが"0"のとき V＝MAC命令の結果、オーバフローが発生したとき

m＝{31：ロングワードサイズのとき
　　15：ワードワードサイズのとき
　　7：バイトワードサイズのとき

Si　：ソースオペランドのビットi
Di　：ディスティネーションオペランドのビットi
Ri　：結果のビットi
Dn　：ディスティネーションオペランドの指定されたビット
－　：影響なし
⇔　：実行結果に応じて変化（定義参照）
0　：常に" 0" クリア
1　：常に" 1" セット
＊　：値を保証しません
Z'　：実行前のZフラグ
C'　：実行前のCフラグ

【０１４６】積和演算を行うＭＡＣ命令、ＭＡＣレジスタをクリアするＣＬＲＭＡＣ命令、汎用レジスタの内容をＭＡＣレジスタに転送するＬＤＭＡＣ命令、ＭＡＣレジスタの内容を汎用レジスタに転送するＳＴＭＡＣ命令がある。

【０１４７】また、汎用レジスタの待避／復帰命令には、１本のレジスタの待避／復帰命令に、ＰＵＳＨ、ＰＯＰ命令が、複数レジスタの待避／復帰命令にＳＴＭ／ＬＤＭ命令がある。ＳＴＭ／ＬＤＭ命令には、指定するレジスタ本数に対応して３種類がある。

【０１４８】図１４に、乗算器２の概略ブロック図を示す。

【０１４９】乗算器２は、入力ラッチ（Ｘ）６１、入力ラッチ（Ｙ）６２、部分積生成回路６３、マツチプレクサ６４、デコーダ６５Ａ、６５Ｂ、６５Ｃ、選択回路６６Ａ、６６Ｂ、６６Ｃ、加算器６７、フィードバック回路６８、乗算結果レジスタ６９などによって構成されている。

【０１５０】乗算器２は１６×１６ビットの乗算を行うことを基本動作とし、さらに、これを利用して、１６×１６ビット＋４２ビットの積和演算を可能としている。

【０１５１】乗算器は乗算動作は、２次のブースのデコードを用いて、１６ビット×６ビットを３回行うようにされる。

【０１５２】１６ビットの乗数Ｙは、Ｙ＝－ｙ［１６］・２＾１５＋Σ（ｙ［ｉ］・２＾（ｉ－１））＝Σ（ｙ［２ｊ］＋ｙ［２ｊ＋１］－２・ｙ［２ｊ＋２］）・２＾２ｊと表現される。ｉ＝１～１５、ｊ＝０～７、ｙ［０］＝０である。

【０１５３】被乗数Ｘとの乗算は、Ｘ・Ｙ＝Σ（ｙ［２ｊ］＋ｙ［２ｊ＋１］－２・ｙ［２ｊ＋２］）・Ｘ・２＾２ｊとなる。ｙ［２ｊ］＋ｙ［２ｊ＋１］－２・ｙ

［２ｊ＋２］は、ｙ［２ｊ］、ｙ［２ｊ＋１］、ｙ［２ｊ＋２］の値の組み合わせにより、０、±１、±２の５種類があるから、部分積（ｙ［２ｊ］＋ｙ［２ｊ＋１］－２・ｙ［２ｊ＋２］）・Ｘは、０、±Ｘ、±２Ｘの５種類である。この内、０、Ｘは直ちに得られる。２Ｘは、１ビットの左シフト（最下位ビットは０）、－Ｘは２の補数であり、論理反転＋１で得られる。－２Ｘは、論理反転＋１の１ビットの左シフト（最下位ビットは０）で得る。

【０１５４】Ｘ側は、採りうる０、±Ｘ、±２Ｘの５種類の部分積（１７ビット）を生成しておく。この５種類を部分積選択回路６６Ａ～６６Ｃに与える。

【０１５５】一方、Ｙ入力のｙ［２ｊ］、ｙ［２ｊ＋１］、ｙ［２ｊ＋２］をデコードして、０、±１、±２を判定して、この結果によって、部分積選択回路６６Ａ～６６Ｃを制御して、前記５種類の部分積を選択する。１回に２ビット単位３種類の選択を行う。これを加算器６７で加算する。加算は、２ビットずつシフトしたそれぞれ１７ビットの部分積を加算して、２２ビット分の結果を得る。不足する上位ビットは符号拡張したデータとする。

【０１５６】この内、下位６ビットは、乗算結果レジスタ６９のビット０～５に格納される。上位１６ビットはフィードバック回路６８を介して、２回めの加算に含められる。２回目の処理では、前記同様に得られた部分積選択回路６６Ａ～６６Ｃの出力である、２ビットずつシフトしたそれぞれ１７ビットの部分積と、１回めの処理の上位１６ビットを加算する。２２ビット分の結果を得る。不足する上位ビットは符号拡張したデータとする。この内、下位６ビットは、乗算結果レジスタ６９のビット１１～６に格納される。上位１６ビットはフィードバック回路６８を介して、２回めの加算に含められる。

【０１５７】同様に３回めの処理が行われる。加算結果下位２０ビットが乗算結果レジスタ６９のビット３１～１２に格納される。加算結果の最上位２ビットは無視する。

【０１５８】積和演算の場合は、前回の結果が同時に加算されるようにする。

【０１５９】更に、前回の結果の上位ビットとの４回目の加算を行って、４２ビットの結果を得る。１６ビット×１６ビットの積和の結果を４２ビットで得ることにより、約１０００回の積和演算を繰り返してもオーバフローしないことになる。

【０１６０】内部論理の構成上は、４０ビットの結果とすれば、加算器が２２ビット長でよく、論理的規模を最適化できる。

【０１６１】図１５に、上記のワードサイズ乗算（１６ビット×１６ビット）の演算方法を示す。

【０１６２】８ビット×８ビットのバイトサイズ乗算は、上位を拡張する。符号無しの場合０拡張、符号付きの場合符号拡張を行う。いずれの場合も、上位は全ビット”０”か全ビット”１”かのいずれかであって、ブースのデコードは０になる。このため、３回目の処理は行わずに、２回の処理で済む。

【０１６３】図４において、ＣＰＵ１から乗算器２に、乗算を示す信号として、ＭＵＬ信号（制御信号Ｂ）、符号付き／無しを示す信号としてＵＮＳＩＮＰ信号（制御信号Ｂ）、バイト／ワードサイズを示す信号として、ＢＹＴＥ信号（制御信号Ｂ）、積和演算の起動信号として、ＭＡＣ信号（制御信号Ｂ）、ＭＡＣＨからＣＰＵへのデータ転送要求信号として、ＳＴＭＡＣＨ信号（制御信号Ｃ）、ＭＡＣＬからＣＰＵへのデータ転送要求信号として、ＳＴＭＡＣＬ信号（制御信号Ｃ）、ＣＰＵからＭＡＣＨへのデータ転送要求信号として、ＬＤＭＡＣＨ信号（制御信号Ｂ）、ＣＰＵからＭＡＣＬへのデータ転送要求信号として、ＬＤＭＡＣＬ信号（制御信号Ｂ）、ＭＡＣレジスタのクリア信号として、ＣＬＲＭＡＣ信号（制御信号Ｂ）、乗数の転送信号として、ＳＴＸ信号（制御信号Ｂ）、被乗数の転送信号として、ＳＴＹ信号（制御信号Ｂ）、乗算結果の転送信号として、ＭＵＬＲＤ信号（制御信号Ｃ）を与える。

【０１６４】また、乗算器２からＣＰＵ１へ、演算実行中を示す信号として、ＢＵＳＹ信号、フラグに反映すべきデータとして、ＶＦＬＡＧ、ＺＦＬＡＧ、ＮＦＬＡＧ信号が与えられる。

【０１６５】ＣＰＵ－乗算器の相互のデータ転送にＸバス、Ｙバスを使用する。

【０１６６】また、ＳＹＳＣＲ１３から飽和演算の選択を示す信号として、ＦＩＸＥＤ信号が与えられる。また、テストモード信号として、ＴＥＳＴＭＯＤＥ信号が与えられる。ＴＥＳＴＭＯＤＥ信号が活性状態になって、テストモードが指示されると、乗算器は１回の処理のみで動作を終了するようにする。

【０１６７】処理を短縮することによって、ＣＰＵの命令実行ステートも短縮できる。入力データの組み合わせを種々変更してテストする場合に、テスト時間を短縮できる。加算を１回しか行わないので、テスト設計を容易にすることができる。３回の処理を行って、加算結果が蓄積されて、所望の動作のテストの結果を演算結果として得にくくなることがない。

【０１６８】ＴＥＳＴＭＯＤＥ信号が非活性状態であっても、ＣＰＵ乃至乗算器のテストを行うことができることは言うまでもない。ＴＥＳＴＭＯＤＥ信号は、前記のＣＰＵＳのようにレジスタの出力として供給することができる。

【０１６９】表２０に乗算器２の内部のフラグの検出方式およびＣＰＵ１への転送方式を示す。

【０１７０】

【表２０】

表２０

ＯＶＦ.ＵＮＦしたときの乗算器のフラグ

条件		Ｖフラグ	Ｎフラグ	Ｚフラグ	演算結果のＭＳＢ
ＯＶＦ/ＵＮＦしないとき		0	N=結果のＭＳＢ	結果=H'００...のときｚ＝1、結果◇H'００...のときＺ＝0	ＭＳＢ＝Ｎフラグ
ＯＶＦしたとき	積和演算	1	1	結果=H'００...のときｚ＝1、結果◇H'００のときＺ＝0	1
	飽和演算	1	0	φ	0
ＵＮＦしたとき	積和演算	1	0	結果=H'００...のときｚ＝1、結果◇H'００のときＺ＝0	0
	飽和演算	1	1	φ	1

結果：　飽和演算で、ＯＶＦ/ＵＮＦしたとき、以下のようにする。
・ＯＶＦしたとき：演算結果＝H'７ＦＦＦＦＦＦＦ、Ｎ＝0、Ｚ＝0にマスクする
・ＵＮＦしたとき：演算結果＝H'８０００００００、Ｎ＝1、Ｚ＝0にマスクする

（演算結果とフラグを一致させる）

【０１７１】乗算器２のフラグ仕様は次のように、①Ｖフラグおよび②Ｎフラグ、Ｚフラグから構成されている。

【０１７２】①Ｖフラグ

セット条件はＭＡＣ命令実行中にオーバフローまたはアンダフローが発生したときである。

【０１７３】クリア条件はＬＤＭＡＣまたはＣＬＲＭＡＣ命令を実行したときである。

【０１７４】乗算器からＣＣＲへの転送は、ＳＴＭＡＣ実行時に行われる。

【０１７５】従って、一連の連続した積和演算中に１回でもオーバフローまたはアンダフローが発生すると、乗算器のＶフラグはセットされた状態を保持する。ＬＤＭＡＣまたはＣＬＲＭＡＣ命令を実行して、新しい一連の積和演算の開始が判断されると、乗算器のＶフラグはクリアされる。

【０１７６】②Ｎフラグ、Ｚフラグ

ＭＵＬ命令用Ｎ、ＺフラグとＭＡＣ命令用Ｎ、Ｚフラグを別々に設けて出力する。

【０１７７】乗算器からＣＣＲへの転送は、乗算（ＭＵＬ）命令の場合、乗算結果の転送時、ＭＡＣ命令の場合ＳＴＭＡＣ実行時に行われる。

【０１７８】なお、ＮフラグとＺフラグは、ＬＤＭＡＣ／ＣＬＲＭＡＣによって変化しない。

【０１７９】図１６にＶフラグ仕様の実現の概念図を示し、図１７にＮフラグ、Ｚフラグ仕様の実現の概念図を示す。

【０１８０】Ｖフラグはセットリセット型のフリップフロップ（ＲＳ－Ｆ／Ｆ）で構成され、一旦、オーバフローまたはアンダフローが発生すると、ＳＴＭＡＣにより読み出すまで状態を保持する。

【０１８１】Ｎ、Ｖフラグはラッチ回路（Ｄ－Ｆ／Ｆ）とマルチプレクサ（ＭＰＸ）で構成される。ＭＡＣ命令実行時の演算結果はラッチ回路に保持され、マルチプレクサに与えられる。また、乗算命令実行時の演算結果は、直接マルチプレクサに与えられる。マルチプレクサはＳＴＭＡＣ命令のときラッチ回路の出力を出力し、それ以外のとき演算結果を直接出力する。

【０１８２】ＭＡＣ命令とその他の命令は並列して動作する。ＭＡＣ命令のフラグを随時ＣＣＲに反映しては、並列実行中の命令のフラグ動作と矛盾してしまう。ＭＡＣ命令のフラグを乗算器内部で保持して、ＳＴＭＡＣ命令実行時にＣＣＲに転送するようにして、上記矛盾を回避することができる。

【０１８３】図１８に、バススイッチ３４のブロック図を示す。

【０１８４】バススイッチ３４は、選択回路７１Ａ、７１Ｂ、拡張回路７２Ａ、７２Ｂ、７２Ｃ、出力バッファ

７３Ａ、７３Ｂ、７３Ｃ、７３Ｄから構成される。バススイッチは、ＣＰＵ内部バスのＧＢ、ＤＢ、ＷＢと、乗算器のＸバス、Ｙバスと、マイクロコンピュータの内部バスであるＩＤＢのインタフェースを行う。

【０１８５】乗算の開始時、及びＬＤＭＡＣ命令の場合は、ＧＢ、ＤＢからＸバス、Ｙバスに入力される。ＧＢ、ＤＢの入力は、選択回路７１Ａ、７１Ｂで選択される。これは汎用レジスタ及び内部バスが３２ビット構成であるために、乗数、被乗数が８ビットまたは１６ビットであるために、ＣＯＮＴ２２の制御信号Ａ及びレジスタ制御信号Ａに基づいて、所定の部分が選択される。

【０１８６】選択回路７１Ａ、７１Ｂの出力は、拡張回路７２Ａ、７２Ｂに入力される。ＣＯＮＴ２２の制御信号Ａに基づいて、符号無しバイトサイズ乗算（ＭＵＬＸＵ．Ｂ）の場合、上位８ビットを０拡張する。また、符号付きバイトサイズ乗算（ＭＵＬＸＳ．Ｂ）の場合、上位８ビットを符号拡張する。ワードサイズの場合は、選択回路７１Ａ、７１Ｂの出力をそのまま出力する。

【０１８７】選択回路７１Ａ、７１Ｂの出力は出力バッファ７３Ｂ、７３Ｃに入力される。ＣＯＮＴ２２の制御信号Ａに基づいて、所定のタイミングで、拡張回路７２Ａ、７２Ｂの出力をＸバスまたはＹバスに出力する。

【０１８８】乗算の終了時、及びＳＴＭＡＣ命令の場合は、Ｘバス、ＹバスからＷＢへの出力が行われる。Ｘバス、Ｙバスの入力は拡張回路７２Ｃに入力される。これは、ＭＡＣＨの上位２２ビットを符号拡張する。拡張回路７２Ｃの出力は出力バッファに入力される。ＣＯＮＴ２２の制御信号Ｃに基づいて、所定のタイミングで、拡張回路７２Ｃの出力をＷＢに出力する。

【０１８９】ＭＡＣ命令のデータリード時、ＩＤＢからＸバス、Ｙバスへの入力が、それぞれ１回ずつ行われる。ＣＯＮＴ２２の制御信号Ａに基づいて、所定のタイミングで、ＩＤＢの内容はをＸバスまたはＹバスに出力する。また、ＩＤＢは、ＤＢＷからの出力を入力可能とされ、ＤＢＲ及びＩＲへデータを入力可能とされる。

【０１９０】図１９、２０に、ＭＡＣ命令の動作タイミングを示す。

【０１９１】例えば、ＭＡＣ　@ＥＲ１＋，@ＥＲ２＋命令などの例である。この場合のＥＲ１を第１のアドレスレジスタ、ＥＲ２を第２のアドレスレジスタとする。前記同様に、Ｔ２からＭＡＣ命令の実行が開始される。

【０１９２】まず、プリフィックスコードの実行を行い、ＰＣ３０の内容をアドレスとした命令のリードを行い、また、ＰＣ３０の内容のインクリメントを行う。

【０１９３】Ｔ３のφ＃で、レジスタ制御信号Ａに基づいて、第１のアドレスの内容をＧＢに読み出して、ＭＡＢ３３に転送し、ＩＡＢに出力する。

【０１９４】Ｔ４のφで、第１のアドレスの内容をＧＢに読み出して、ＡＬＵ２６に入力し、インクリメントを行う。

【０１９５】Ｔ４のφ＃で、インクリメント結果を、ＷＢ経由で、第１のアドレスレジスタに格納する。バススイッチ３４に入力した、第１のリードデータをＸバスに出力すると共に、制御信号Ｂに含まれるＳＴＸ信号を活性状態にし、乗算器２にこの内容を入力ラッチＸにラッチさせる。同時に、第２のアドレスの内容をＧＢに読み出して、ＭＡＢに転送し、ＩＡＢに出力する。

【０１９６】Ｔ５のφで、第２のアドレスの内容をＧＢに読み出して、ＡＬＵ２６に入力し、インクリメントを行う。Ｔ５のφ＃で、インクリメント結果を、ＷＢ経由で、第２のアドレスレジスタに格納する。バススイッチ３４に入力した、第２のリードデータをＹバスに出力すると共に、制御信号Ｂに含まれるＳＴＹ信号を活性状態にし、乗算器２にこの内容を入力ラッチＹにラッチさせる。ＭＡＣ信号を活性状態にして、積和演算動作の開始を指示する。同時に、ＰＣ３０の内容をアドレスとした命令のリードを行い、また、ＰＣ３０の内容のインクリメントを行う。

【０１９７】Ｔ６のφで、インクリメント結果をＰＣ３０に格納する。一方、ＢＵＳＹ信号が活性状態になる。ＭＡＣ命令では、ＣＰＵ１は乗算器２とは並列に動作し、ＢＵＳＹ信号を無視して、次の命令の実行を開始する。

【０１９８】ＭＡＣ命令を連続して実行した場合も、次のＭＡＣ命令がアドレス計算を行っている間に、乗算器の動作が終了するために、ＭＡＣ命令実行にウェイトを挿入することはない。

【０１９９】プリフィックスコードを付した命令コードとすることにより、特開平－５１９８１号公報に記載されているように、互換性を保持しつつ命令セットを拡張することができる。また、乗算器動作中に、次の積和演算を行った場合、命令フェッチとデータのアクセスを行うことができるから、命令長が長くなっても実行時間を低下させることがない。積和演算を連続的に高速に実行することができる。

【０２００】乗算器２は、演算終了時点で、ＳＹＳＣＲ１３のＭＡＣＳビットを参照して、オーバフローが発生していれば、ＭＡＣレジスタの内容を、上限（Ｈ’７ＦＦＦＦＦＦＦ）または下限（Ｈ’８０００００００）に固定する。

【０２０１】図２１、２２に、ＳＴＭＡＣ、ＬＤＭＡＣ命令の動作タイミングを示す。

【０２０２】例えば、ＳＴＭＡＣ　ＭＡＣＨ，ＥＲ２命令などの例である。前記同様に、Ｔ２からＳＴＭＡＣ命令の実行が開始される。

【０２０３】まず、ＢＵＳＹ信号の状態をサンプリングする。ＢＵＳＹ信号が活性状態であれば、ウェイト状態になる。

【０２０４】Ｔ２のφ＃でＰＣの内容がＧＢに読み出され、ＭＡＢ３３に入力されて、ＩＡＢに出力される。ま

た、ＩＮＣ２７に入力されて、インクリメント動作が開始される。

【０２０５】ＣＰＵ内部のクロックがロウレベルで固定され、ＣＰＵの動作を停止する。直前にＭＡＣ命令を実行した場合、ＢＵＳＹ信号は３ステートの期間活性状態であり、ＳＴＭＡＣ命令も３ステートウェイト状態になる。

【０２０６】Ｔ５でＢＵＳＹ信号が非活性状態になると、Ｔ６からクロックの動作が開始される。

【０２０７】Ｔ６のφで、インクリメント結果がＷＢに出力され、ＰＣ３０に格納される。ＳＴＭＡＣＨまたはＳＴＭＡＣＬ信号が活性状態になって、ＭＡＣレジスタの読み出しが指示される。ＭＡＣレジスタの内容がＸバス、Ｙバスに出力される。特に制限はされないものの、Ｘバスが上位、Ｙバスが下位の内容とされる。

【０２０８】Ｔ６のφ＃で、Ｘバス、Ｙバスの内容がＷＢに出力されて、指定されたレジスタ（ＳＴＭＡＣ　ＭＡＣＨ，ＥＲ２の場合は、ＥＲ２）に格納される。同時に、乗算器のフラグの内容がＣＣＲのＮ、Ｚ、Ｖフラグに格納される。

【０２０９】また、ＬＤＭＡＣ　ＥＲ１，ＭＡＣＬ命令などの例である。

【０２１０】Ｔ８からＬＤＭＡＣ命令の実行が開始される。

【０２１１】Ｔ９のφで、指定されたレジスタ（ＬＤＭＡＣ　ＥＲ１，ＭＡＣＬの場合は、ＥＲ１）の内容が読み出される。この内容がＸバス、Ｙバスに出力される。

【０２１２】Ｔ９のφ＃で、ＬＤＭＡＣＨまたはＬＤＭＡＣＬ信号が活性状態になる。ＰＣ３０の内容がＧＢ経由で、ＭＡＢ３３とＩＮＣ２７に入力される。

【０２１３】Ｔ１０のφで、インクリメントされた結果がＷＢ経由で、ＰＣ３０に格納される。また、Ｘバス、Ｙバスの内容がＭＡＣレジスタに格納される。

【０２１４】前記同様に、ＢＵＳＹ信号が活性状態の場合は、ＬＤＭＡＣ命令も活性状態になるようにしてもよい。

【０２１５】ＣＬＲＭＡＣ命令は、概略ＬＤＭＡＣ命令と同様の動作で、ＬＤＭＡＣ命令のＬＤＭＡＣＨ、Ｌ信号と同じタイミングで、ＣＬＲＭＡＣ信号を活性状態にするようにすればよい。

【０２１６】図２３、２４に、乗算器を用いた乗算命令のタイミング図を示す。

【０２１７】なお、ＣＯＮＴ２２の部分に、内部のステートマシンのステップの番号を記載した。これは、基本的には、ＣＯＮＴ２２の出力のフィードバック信号で形成される。また、制御信号ＣＰＵＳを用いて、乗算器を使用するか使用しないかを選択する。例えば、ＭＵＬＸＵ．Ｗ　Ｒ１，ＥＲ０などのバイトサイズ・符号無し乗算の例である。前記同様に、Ｔ２から実行を開始する。

【０２１８】命令が解読されると、まず、Ｔ２のφ＃で、レジスタ制御信号Ａによって、汎用レジスタの読みだしを指示する。読出された結果は、ＧＢ、ＤＢおよびバススイッチ３４を介して、Ｘバス、Ｙバスに出力される。

【０２１９】制御信号Ｂに含まれるＳＴＸ、ＳＴＹ信号に基づいて、Ｘバス、Ｙバスの内容は、Ｔ３のφで乗算器の入力ラッチにラッチされる。また、同時に、制御信号Ｂに含まれるＭＵＬ信号によって、乗算器に乗算を指示する。ＣＯＮＴ２２から、バイト／ワードの選択、符号付／符号無の選択、乗算／積和の選択を上記制御信号によって指示する。

【０２２０】乗算器は、Ｔ３のφ＃で、ＢＵＳＹ信号を与える。また、マルチプレクサやデコーダを動作させる。Ｔ４のφで、１回目の加算を行う。Ｔ４のφで部分積を乗算結果レジスタとフィードバックラッチに格納する。これを３回繰り返す。ＢＵＳＹ信号が活性状態になったことに呼応して、ＣＰＵはウェイト状態になる。

【０２２１】Ｔ５のφでＢＵＳＹが非活性状態になって、ＣＰＵは動作を再開し、Ｔ６のφで、制御信号Ｃに含まれる、ＭＵＬＲＤ信号を活性状態にして、乗算結果レジスタのリードを指示する。乗算結果レジスタの内容は、Ｘバス、Ｙバスおよびバススイッチ３４を経由して、Ｔ６のφ＃でＷＢを経由して、レジスタ制御信号Ｃによって指定されるレジスタに格納される。同時に、乗算の結果フラグがＣＣＲ３１に格納される。

【０２２２】前記の通り、乗算命令はＢＵＳＹ信号によって、クロックが停止し、ウェイト状態となる。バイトサイズ符号無し乗算命令（ＭＵＬＸＵ．Ｂ　Ｒ０Ｌ，Ｒ１など）は１ウェイトが挿入され、３ステートで実行される。ワードサイズ符号無し乗算命令（ＭＵＬＸＵ．Ｗ　Ｒ０，ＥＲ１など）は２ウェイトが挿入され、４ステートで実行される。なお、符号付き乗算の場合は、それぞれ、プリフィックスコードの実行が付加される。

【０２２３】ＢＵＳＹ信号によって、演算実行の終了を判定することにより、制御回路（ＣＯＮＴ２２）の論理を縮小することができる。

【０２２４】ＴＥＳＴＭＯＤＥ信号が活性状態になって、テストモードを指示された場合には、乗算器は１ステップの動作のみを行い、ＢＵＳＹ信号は非活性状態を保持する。ＣＰＵは１ステートで処理を終了する。

【０２２５】図２５、２６に、乗算器を用いない乗算命令のタイミング図を示す。乗算器を用いない乗算は、特に制限はされないものの、除算と類似のシーケンスで行うようにする。

【０２２６】命令が解読されると、まず、汎用レジスタの読みだしを指示する。読出された結果は、符号判定を行う。符号付／符号無の選択に対応して、符号判定を行い、除数は符号反転し、負数にする。そのほかは正数にする。

【０２２７】被乗数を上位、下位は０にして１ビットず

つシフトし、シフトした結果によって、下位側に乗数を加算するかを決める。その結果に対して、さらにシフトを行い、シフトした結果によって、下位側に乗数を加算を行っていく。これを８または１６回繰返して、乗算結果の絶対値を得る。例えば、ＭＵＬＸＵ．Ｂ　Ｒ１Ｌ，Ｒ０などのバイトサイズ・符号無し乗算の例である。前記同様に、Ｔ２から除算命令の実行が開始される。前記のような、所定の処理を行った後、Ｔ５から部分乗算を行う。

【０２２８】部分乗算は、左シフト処理と加算で構成される。前回の加算と次回のシフト処理を同一のＡＬＵ処理で行うようにする。

【０２２９】Ｔ５のφ１に同期して、指定されたレジスタ（ディスティネーションレジスタＲｄ）から被乗数を読み込み、シフト処理を行う。シフト処理の結果（部分積）がφ＃に同期してＷＢを経由して、Ｒｄにライトされる。また、シフトアウトされたキャリが内部で保持される。

【０２３０】Ｔ６のφ１に同期して、Ｒｄから部分積を読み込み、部分乗算処理を行う。前回のキャリが”１”である場合、部分積の下位８ビットに乗数を加算し、１６ビットでシフト処理を行い、最下位ビットは”０”とする。前記以外の場合、部分積に１６ビットでシフト処理を行い、最下位ビットは”０”とする。かかる結果がφ＃に同期してＷＢを経由して、Ｒｄにライトされる。また、シフトアウトされたキャリが内部で保持される。この動作を７回繰り返す。

【０２３１】Ｔ１３では、上記同様の判定を行い、前回のキャリが”１”である場合、部分積の下位８ビットに乗数を加算する。前記以外の場合、部分積を保持する。１６ビットの積が得られる。かかる結果がφ＃に同期してＷＢを経由して、Ｒｄにライトされる。符号付きの場合は、Ｔ１４で符号処理を行う。また、ワードサイズの場合は、部分乗算処理が８回追加される。

【０２３２】先に保持した符号判定結果に基づいて、積の符号処理を行う。すなわち、乗数・被乗数の一方が正数、他方が負数のときは、積の符号を反転する（０から積を引く）。

【０２３３】図２７に、乗算命令の状態遷移図を示す。例えば、ＭＵＬＸＵ．Ｂ　Ｒ１Ｌ，Ｒ０などのバイトサイズ・符号無し乗算の例である。命令の実行が開始されると、ＣＰＵＳ信号の状態によって分岐する。

【０２３４】ＣＰＵＳ信号が活性状態であって、乗算器の使用が許可されると、図６の動作を行う。即ち、ステップ１で、指定されたレジスタの内容を、ＧＢ、ＤＢを経由して、Ｘ、Ｙバスに出力して、乗算器に供給する。ＢＵＳＹ信号の状態を判定する。テストモードであれば、ＢＵＳＹ信号は非活性状態であって、直ちにステップ２に遷移する。

【０２３５】ＢＵＳＹ信号が活性状態であると、ＷＡＩＴ状態に遷移する。ＢＵＳＹ信号は非活性状態になるとステップ２に遷移する。

【０２３６】ステップ２では、Ｘ、Ｙバスの内容をＷＢを経由して、指定されたレジスタにライトする。例えば、乗算器のフラグの内容をＣＣＲ３１に格納する。次の命令の実行を開始する。

【０２３７】ＣＰＵＳ信号が非活性状態であって、乗算器の使用が禁止されると、図２５、２６の動作を行う。即ち、ステップ１、２でデータアライメントなどを行った後、ステップ３から、部分乗算処理を行う。ステップ３では、ＧＢ上位に被乗数を出力し、これをシフトする。

【０２３８】ステップ４では、ＧＢに部分積を、ＤＢ下位に乗数を出力し、ＡＬＵ２６で加算を行う。前のステップでシフトアウトしたビットが”１”であれば、加算した結果が選択され、シフトアウトしたビットが”０”であれば、ＧＢの内容が選択され、シフトを行う。これをステップ１０まで繰り返す。

【０２３９】ステップ１１では、ＧＢに部分積を、ＤＢ下位に乗数を出力し、ＡＬＵ２６で加算を行う。前のステップでシフトアウトしたビットが”１”であれば、加算した結果が選択され、シフトアウトしたビットが”０”であれば、ＧＢの内容が選択される。シフトは行わない。

【０２４０】ステップ１２で、命令のリードを行う。例えば、積を検査して、ＣＣＲに反映する。次の命令の実行を開始する。

【０２４１】図２５、２６に、除算命令のタイミング図を示している。例えば、ＤＩＶＸＵ．Ｂ　Ｒ１Ｌ，Ｒ０などのバイトサイズ・符号無し除算の例である。前記同様に、Ｔ２から除算命令の実行が開始される。除数の符号反転などの、所定の処理を行った後、Ｔ５から部分除算を行う。部分除算は、左シフト処理と減算で構成される。前回の減算と次回のシフト処理を同一のＡＬＵ処理で行うようにする。

【０２４２】Ｔ５のφ１に同期して、指定されたレジスタ（ディスティネーションレジスタＲｄ）から被除数を読み込み、シフト処理を行う。シフト処理の結果（部分剰余）がφ＃に同期してＷＢを経由して、Ｒｄにライトされる。また、シフトアウトされたキャリが内部で保持される。

【０２４３】Ｔ６のφ１に同期して、Ｒｄから部分剰余を読み込み、部分除算処理を行う。前回のキャリが”１”である場合、または、部分剰余の上位８ビットが除数以上である場合、部分剰余の上位８ビットから除数を減算（除数の符号反転を行っている場合、除数の反転を加算）し、１６ビットでシフト処理を行い、最下位ビットは”１”とする。前記以外の場合、部分剰余に１６ビットでシフト処理を行い、最下位ビットは”０”とする。かかる結果がφ＃に同期してＷＢを経由して、Ｒｄ

にライトされる。また、シフトアウトされたキャリが内部で保持される。この動作を７回繰り返す。

【０２４４】Ｔ１３では、上記同様の判定を行い、前回のキャリが”１”である場合、または、部分剰余の上位８ビットが除数以上である場合、部分剰余の上位８ビットから除数を減算し、下位８ビットでシフト処理を行い、最下位ビットは”１”とする。前記以外の場合、部分剰余に下位８ビットでシフト処理を行い、最下位ビットは”０”とする。いずれの場合も、ビット７の値は失われる。上位８ビットに剰余、下位８ビットに商が得られる。かかる結果がφ＃に同期してＷＢを経由して、ＲｄにライトされるＲｄにライトされる。

【０２４５】符号付きの場合は、Ｔ１４で符号処理を行う。また、ワードサイズの場合は、部分除算処理が８回追加される。

【０２４６】図２８に、ＡＬＵ２６の概略ブロック図を示す。ＡＬＵ２６は、算術論理演算回路７６と、選択回路７７、シフト回路７８、制御回路７９から構成される。乗除算に直接関係のない部分は省略している。

【０２４７】算術論理演算回路７６は、ＧＢとＤＢの内容を入力して、加算、減算、論理積、論理和、排他的論理和などの演算を行い、結果を出力する。選択回路７７は、算術論理演算回路の出力と、ＧＢの内容を入力して、いずれかを選択して出力する。シフト回路７８は、選択回路７７の出力を入力して、シフト処理を行う。

【０２４８】選択回路７７、シフト回路７８は制御回路７９によって制御される。制御回路７９は、ＣＯＮＴ２２の与える制御信号と算術論理演算回路７６とシフト回路７８の出力によって、選択回路７７の選択とシフト回路７８のシフト入力を制御する。制御回路７９が、前記の部分乗算、部分除算の判定を行う。条件が成立していれば、算術論理演算回路７６の出力を選択し、除算の場合、１をシフト回路に入力する。条件が不成立であれば、ＧＢの入力を選択し、除算の場合、０をシフト回路に入力する。乗算の場合のシフト回路の入力は、０とされる。

【０２４９】除算の部分除算と乗算の部分乗算の処理のシーケンス、及びＡＬＵ２６の回路構成を共通化する。除算と乗算を共通化して、ＣＯＮＴ２２の論理規模を縮小できる。

【０２５０】これにより、乗算器を持たないＣＰＵを容易に提供することができる。乗算器を持つＣＰＵにおいて不必要な乗算器を用いない乗算の論理を除算と共通化して、論理規模の増加を最低限にすることができる。

【０２５１】また、ＣＰＵＳによって、乗算器を用いない選択を可能にすることによって、テスト性を向上することができる。テスト時に、乗算器を用いるか用いないかを選択することに両方の論理をテストの対象にすることができる。

【０２５２】複数命令の待避／復帰命令の命令コードは表１６の通りである。

【０２５３】最初に使用するレジスタ番号が、命令コード中に指定される。例えば、昭和５年３月（株）日立製作所発行『Ｈ８／５００シリーズプログラミングマニュアル』に記載の複数命令の待避／復帰命令のように任意のレジスタの組合わせを指定するのではなく、連続したレジスタ番号の固定の組合わせとし、２、３、４本の固定の組合わせとしている。命令コードも、レジスタ本数に応じて３種類を用意している。

【０２５４】複数命令の待避命令は、待避するレジスタの本数に対応して、

ＳＴＭ（ＥＲｌ－ＥＲｌ＋１），@－ＳＰ
ＳＴＭ（ＥＲｍ－ＥＲｍ＋２），@－ＳＰ
ＳＴＭ（ＥＲｎ－ＥＲｎ＋３），@－ＳＰ

の３種類を有する。ｌ＝０、２、４、６であり、ｍ、ｎ＝０、４である。指定した汎用レジスタをスタックに待避する。例えば、ＥＲ０とＥＲ１をスタックに待避する場合は、

ＳＴＭ（ＥＲ０－ＥＲ１），@－ＳＰ

を用いる。ＥＲ０、ＥＲ１の順番でスタックにライトされ、スタックポインタ（ＥＲ７）は＋８される。命令コード中のレジスタ指定部は、最初に待避されるレジスタ番号にしてある。

【０２５５】図２９、３０に複数レジスタの待避命令の実行シーケンスを示す。例えば、ＳＴＭ．Ｌ　ＥＲ０－ＥＲ１，@－ＳＰなどの２本の汎用レジスタを待避する例である。レジスタ指定フィールドはＢ’０００である（Ｂ’は２進数を示す）。

【０２５６】前記同様に、Ｔ２から除算命令の実行が開始される。特に制限はされないものの、命令コードの第１ワードはプリフィックスコードであり、次の命令コードの動作を指定し、ＰＣをインクリメントするほかの動作は行わない。

【０２５７】第２ワードの命令コードは、ＰＵＳＨ命令と共通にされる。

【０２５８】Ｔ４のφで、ＳＰの内容をＧＢに読み出し、ＡＬＵ２６に入力する。ＡＬＵ２６では－４の演算を行う。なお、前記の通り、実行前のＳＰはスタックの先頭アドレスを示しているとする。

【０２５９】Ｔ４のφ＃で演算結果がＷＢとＧＢに出力される。ＷＢからＳＰに書き込まれ、ＧＢからＭＡＢ３３に格納される。ＭＡＢ３３の内容がＩＡＢに出力される。また、第１の制御信号ＢとＲＳＥＬ２（＝Ｂ’０００）とによって、待避されるレジスタが選択され、レジスタ制御信号Ｂが生成される。

【０２６０】Ｔ５のφで、選択されたレジスタ（ＥＲ０）の内容がＤＢ経由で、ＤＢＷ２４に転送される。

【０２６１】Ｔ５のφ＃で、転送されたデータ（ＥＲ０の内容）の上位１６ビット（Ｅレジスタの内容）が内部データバスに出力される。また、ＭＡＢ３３のインクリ

メント機能によって、ＩＡＢの出力値を＋２とする。

【０２６２】Ｔ６のφで、更に、ＳＰの内容をＧＢに読み出し、ＡＬＵ２６に入力する。ＡＬＵ２６では－４の演算を行う。

【０２６３】Ｔ６のφ＃で、ＤＢＷ２４に転送されたデータの下位１６ビット（Ｒレジスタの内容）が内部データバスに出力される。ＡＬＵ２６の演算結果がＷＢとＧＢに出力される。ＷＢからＳＰに書き込まれ、ＧＢからＭＡＢに格納される。ＭＡＢ３３の内容がＩＡＢに出力される。また、第２の制御信号ＢによってＲＳＥＬのビット０が反転される。第１の制御信号とＲＳＥＬ２（＝Ｂ’００１）とによって、待避されるレジスタが選択され、レジスタ制御信号Ｂが生成される。

【０２６４】Ｔ７のφで、選択されたレジスタ（ＥＲ１）の内容がＤＢ経由で、ＤＢＷ２４に転送される。

【０２６５】Ｔ７のφ＃で、転送されたデータ（ＥＲ１の内容）の上位１６ビット（Ｅレジスタの内容）が内部データバスに出力される。また、ＭＡＢ３３のインクリメント機能によって、ＩＡＢの出力値を＋２とする。

【０２６６】Ｔ８のφ＃で、ＤＢＷ２４に転送されたデータの下位１６ビット（Ｒレジスタの内容）が内部データバスに出力される。

【０２６７】Ｔ８のφ＃以降で、前記同様に、次の次の命令の読み出しと、ＰＣ３０のインクリメント（＋２）を行う。

【０２６８】レジスタ３本を指定した場合は、実行ステート数が２ステート長くなり、ＳＰのデクリメント（－４）と、ＲＳＥＬのビット１が反転される。ＲＳＥＬは、レジスタ指定フィールドが０００の場合、０１０とされ、汎用レジスタＥＲ２が選択される。ライト動作が２回行われる。

【０２６９】レジスタ４本を指定した場合は、更に、実行ステート数が２ステート長くなり、ＳＰのデクリメント（－４）と、ＲＳＥＬのビット１とビット０が反転される。ＲＳＥＬは、レジスタ指定フィールドが０００の場合、０１１とされ、汎用レジスタＥＲ２３が選択される。ライト動作が２回行われる。

【０２７０】レジスタ番号の下位ビットが固定であるので、これを命令処理の実行に従って、変更させることが容易である。例えば、２本のレジスタを待避する場合、命令コード上のレジスタ指定フィールドの下位ビットは０であるので、１回めのレジスタ指定は、レジスタ指定フィールドの値に従い、２回のレジスタ指定は、ＣＯＮＴ２２の制御に従って、レジスタ指定フィールドの下位１ビットを１に変更して、行うようにする。

【０２７１】一方、ＰＵＳＨ命令はレジスタ１本の待避であり、前記の２回目の待避動作を行わないようにされ、実行動作の共通化を図っている。

【０２７２】図３１、３２に複数レジスタの復帰命令の実行シーケンスを示す。例えば、ＬＤＭ．Ｌ　@ＥＲ７＋，ＥＲ０－ＥＲ１などの２本の汎用レジスタを待避する例である。レジスタ指定フィールドは００１である。

【０２７３】図３３、３４に、ＲＳＥＬ２入力制御回路の具体的な構成、およびその動作説明を示す。この制御回路は、アンド回路７５Ａ、７５Ｂ、オア回路８０Ａ、８０Ｂから構成される。

【０２７４】ビット２には、オペコードのレジスタ指定フィールドのビット２がそのまま入力される。ビット１、０には、オアゲートとアンドゲートを介して入力される。オアゲートの他方の入力はＳＴＭ制御信号１、０であり、アンドゲートの他方の入力はＬＤＭ制御信号１、０の反転とされる。ＳＴＭ制御信号１、０およびＬＤＭ制御信号１、０は、ＣＯＮＴ２２の出力である制御信号Ｂに含まれる。

【０２７５】ＳＴＭ制御信号が活性状態になると、当該ＲＳＥＬビットは１になる。また、ＬＤＭ制御信号が活性状態になると、当該ＲＳＥＬビットは０になる。ＳＴＭ、ＬＤＭ命令と指定したレジスタ本数に従って、ＳＴＭ制御信号、ＬＤＭ制御信号が生成される。

【０２７６】これにより、レジスタ選択回路をそのほかの命令と共通化することができる。共通化によって、物理的規模の増加を抑止できる。

【０２７７】図３５、３６に、Ｃ言語で書かれた関数と、これをＣＰＵの命令に変換したリストの概略を示す。このリストには、オフセット（相対アドレス）、命令コード、Ｃラベル、Ｃソース及びアセンブラ命令の各項目が示されている。

【０２７８】Ｃ言語からＣＰＵの命令へのコンパイルについては、例えば、平成４年９月（株）日立製作所発行『Ｈ８／３００シリーズＣコンパイラ』に記載されている。引数を汎用レジスタＥＲ０、ＥＲ１に設定しておくことができる。

【０２７９】関数Ｐｒｏｃ１では、引数をレジスタ渡しとし、これをＥＲ０に割り当てている。関数内の処理で、ＥＲ２、３、４、６を使用するため、関数処理の先頭で、

ＳＴＭ　（ＥＲ２－ＥＲ３），@－ＳＰ
ＳＴＭ　（ＥＲ４－ＥＲ６），@－ＳＰ

を実行して、関数の最後で、

ＬＤＭ　@ＳＰ＋，（ＥＲ４－ＥＲ６）
ＬＤＭ　@ＳＰ＋，（ＥＲ２－ＥＲ３）

を実行して、サブルーチンからリターン（ＲＴＳ）している。

【０２８０】ＥＲ０、ＥＲ１は引数領域のため、関数内では使用せず、内容の待避／復帰も行わない。

【０２８１】また、この関数内で呼び出される関数Ｐｒｏｃ３は、引数をレジスタ渡しとし、これをＥＲ０に割り当てている。関数内の処理で、ＥＲ５を使用するため、関数処理の先頭で、１レジスタの待避

ＰＵＳＨ．Ｌ　ＥＲ５

を実行して、関数の最後で、
ＰＯＰ．Ｌ　ＥＲ５
を実行して、サブルーチンからリターン（ＲＴＳ）している。

【０２８２】スタックポインタはＥＲ７と兼用であるから、ＥＲ７を待避／復帰することは意味がない。従って、タスク切替えを行う場合に使用可能なすべてのレジスタを待避する場合には、
ＳＴＭ　＠ＳＰ＋，（ＥＲ０－ＥＲ３）
ＳＴＭ　＠ＳＰ＋，（ＥＲ４－ＥＲ６）
の２命令を用いる。ＥＲ０からＥＲ６の順番でスタックに待避される。同様に、復帰する場合には、
ＬＤＭ　＠ＳＰ＋，（ＥＲ４－ＥＲ６）
ＬＤＭ　＠ＳＰ＋，（ＥＲ０－ＥＲ３）
の２命令を用いる。ＥＲ６からＥＲ０の順番でスタックから復帰される。

【０２８３】前記のように任意の組合わせを指定できないが、予め、レジスタの割当てを行っておくことにより、実質的な制約にはなりにくい。７本のレジスタを待避／復帰する場合に２命令を用いることになるが、全体的な実行ステート数やプログラム容量に対しては影響が小さい。少なくとも、１本のレジスタずつの待避／復帰命令を用いるより効果がある。後者の場合、４バイト×７、５ステート×７であるのに対して、前者では、４バイト×２、９＋１１ステートで実行できる。少なくとも、命令リードのためのリードサイクルや、アドレス計算のための内部動作の分を短縮して、高速化を図ることができる。

【０２８４】前記Ｃ言語で書かれたプログラムのように、関数乃至サブルーチンを多く用いるプログラムの高速化を実現することができる。

【０２８５】また、上記のような関数乃至サブルーチンの場合のほかに、割り込み処理ルーチンにおいても、同様のレジスタの待避／復帰を行う必要がある。マイクロコンピュータが機器制御などを行う場合には、割り込み処理については、割り込みのイベントが発生してから、実際の割り込み処理を実行するまでの時間を短縮することによって、リアルタイム制御性を向上することができる。複数レジスタの待避を高速に実行可能にすることにより、かかるリアルタイム制御性の向上に効果がある。

【０２８６】また、固定の組合わせにし、各命令の実行ステート数を固定にすることにより、内部の条件分岐を行うことをなくし、内部論理を簡潔にし、論理規模を縮小できる。マイクロプログラムによらず、ワイアードロジックなどでも容易に実現できる。マイクロプログラムによらず、ワイアードロジックなどとすることにより、論理回路の高速化に寄与することができる。特に、Ｃ言語など関数乃至サブルーチンを多く用いるプログラムを高速に実行することができる。

【０２８７】図３７、３８に、割込み例外処理のシーケンスを示す。

【０２８８】図３９に、例外処理の状態遷移図を示す。

【０２８９】前記同様に、Ｔ２から割り込み例外処理の実行が開始される。プリフェッチした命令はキャンセルされ、図示されない割り込み要求信号に呼応して、ＣＯＮＴ２２の入力が切り替えられる。

【０２９０】ステップ１の動作として、ＰＣ３０のデクリメントを行う。Ｔ２のφ＃で、ＰＣ３０の内容を読み出して、ＧＢ経由で、ＩＮＣ２７でデクリメント（－４）を行う。これはプリフェッチをキャンセルしたことに対応して、待避すべきＰＣ３０の値を算出する。

【０２９１】Ｔ３のφで、デクリメントした結果を、ＷＢ経由で一旦ＰＣ３０に格納する。

【０２９２】ステップ２で、ＳＰをデクリメントし、この内容をアドレスとして、ＰＣ３０の内容をデータとして、ライト動作を行う。即ち、Ｔ３のφで、同時に、ＳＰの内容を読み出して、ＧＢ経由でＡＬＵ２６でデクリメント（－２）を行う。

【０２９３】Ｔ３のφ＃で、デクリメントした結果を、ＷＢ経由でＳＰに格納するとともに、ＧＢ経由でＭＡＢ３３に転送し、ＩＡＢに出力させる。

【０２９４】Ｔ４のφで、ＰＣ３０の内容をＤＢ経由でＤＢＷ２４に転送する。ＤＢＷ２４の内容は、Ｔ４のφ＃から、内部データバスに出力される。

【０２９５】ステップ３で、ＳＰをデクリメントし、この内容をアドレスとして、ＰＣ３０の上位８ビットとＣＣＲ３１の内容をデータとして、ライト動作を行う。

【０２９６】Ｔ４のφで、同時に、ＳＰの内容を読み出して、ＧＢ経由でＡＬＵ２６でデクリメント（－２）を行う。

【０２９７】Ｔ４のφ＃で、デクリメントした結果を、ＷＢ経由でＳＰに格納するとともに、ＧＢ経由でＭＡＢ３３に転送し、ＩＡＢに出力させる。ＤＢＷ２４に保持したＰＣ３０の内容下位１６ビットを内部データバスに出力する。

【０２９８】Ｔ５のφで、ＣＣＲ３１の内容をＤＢ経由でＤＢＷ２４に転送する。Ｔ４で格納したＰＣ３０の上位８ビットは保持される。ＤＢＷ２４の内容は、Ｔ５のφ＃から、内部データバスに出力される。

【０２９９】ＩＮＴＭ１信号が非活性状態であれば、ステップ４に遷移する。ＩＮＴＭ１信号が活性状態であれば、ステップ１２に遷移し、ＳＰをデクリメントし、この内容をアドレスとして、ＥＸＲの内容をデータとして、ライト動作を行う。

【０３００】Ｔ５のφで、ＳＰの内容を読み出して、ＧＢ経由でＡＬＵ２７でデクリメント（－２）を行う。

【０３０１】Ｔ５のφ＃で、デクリメントした結果を、ＷＢ経由でＳＰに格納する。

【０３０２】Ｔ６のφで、ＥＸＲの内容をＤＢ経由でＤＢＷ２４に転送する。Ｔ６のφ＃から、内部データバス

に出力される。

【０３０３】ステップ４で、ベクタアドレスの内容をリードする。

【０３０４】Ｔ５のφ＃で、同時に、ＶＡＧの内容をＧＢ経由でＭＡＢ３３に転送し、ＩＡＢに出力させる。ＶＡＧには、図示されない、割り込みコントローラから与えられるベクタ番号に基づいて、ベクタアドレスを生成する。

【０３０５】ステップ５で、ベクタアドレスのリード動作の終了を待つ。

【０３０６】ステップ６で、ＤＢＲ２５に格納した、ベクタアドレスの内容をアドレスとして、命令のリードを行う。ＤＢＲ２５の内容をインクリメントし、ＰＣ３０に格納する。

【０３０７】Ｔ８のφ＃で、ＤＢＲ２５に格納したベクタアドレスのリード内容（分岐先の先頭アドレス）をＧＢ経由で、ＭＡＢ３３に転送し、ＩＡＢに出力させ、ＡＬＵ２６でインクリメント（＋２）する。

【０３０８】Ｔ９のφで、インクリメントした結果を、ＷＢ経由でＰＣ３０に格納する。

【０３０９】ステップ７で、ＰＣ３０の内容をアドレスとして、命令のリードを行い、ＰＣ３０のインクリメントを行う。

【０３１０】Ｔ９のφ＃で、ＰＣ３０の内容（分岐先の先頭アドレス）をＧＢ経由で、ＭＡＢ３３に転送し、ＩＡＢに出力させ、ＡＬＵ２６でインクリメント（＋２）する。リードした命令をＩＲ２１に格納する。

【０３１１】Ｔ１０のφで、インクリメントした結果を、ＷＢ経由でＰＣ３０に格納する。

【０３１２】次の命令の実行を開始させる。

【０３１３】制御信号ＩＮＴＭ１に従って、ステップ１２を行うか、行わないかが選択され、スタックを２回行うか、３回行うかが選択される。スタックを２回行う場合には、ＰＣとＣＣＲ３１のみが待避される。ＳＰは－４となる。上記Ｔ５の動作に相当する部分（ステップ１２）が実行されない。３回行う場合には、ＰＣ３０とＣＣＲ３１及びＥＸＲが待避される。ＳＰは－６となる。

【０３１４】なお、ステップ１単位の動作が複数ステートにまたがっているのは、１つのＣＯＮＴ２２の入力に対応して、複数の異なるタイミングの制御信号Ａ、Ｂ、Ｃ及びレジスタ選択信号Ａ、Ｂ、Ｃが生成されるのに対応する。

【０３１５】図４０、４１に、例外処理後のスタックの状態を示す。図４１はノーマルモードを示し、図４２はアドバンストモードを示している。

【０３１６】図４２に、ＲＴＥ命令の実行シーケンスを示す。

【０３１７】図４３に、例外処理の状態遷移図を示す。

【０３１８】前記同様に、Ｔ２からＲＴＥ命令の実行が開始される。

【０３１９】ステップ１の動作として、ＳＰの内容をアドレスとして、スタックのリードを行う。Ｔ２のφ＃で、ＳＰの内容を読み出して、ＧＢ経由で、ＭＡＢ３３に転送し、ＩＡＢに出力させる。

【０３２０】Ｔ３のφで、ＳＰの内容を読み出して、ＧＢ経由で、ＡＬＵ２７でインクリメント（＋２）する。ＩＡＢのアドレスでスタックをリードする。Ｔ３のφ＃で、リードした内容をＤＢＲ２５に格納する。

【０３２１】ＩＮＴＭ１信号が非活性状態であれば、ステップ２に遷移する。ＩＮＴＭ１信号が活性状態であれば、ステップ１０に遷移し、リードした結果をＥＸＲに格納する。ＳＰをインクリメントし、この内容でリードを行う。

【０３２２】Ｔ３のφ＃で、同時に、インクリメントした結果を、ＷＢ経由でＳＰに格納する。また、ＧＢを経由して、ＭＡＢ３３に転送し、ＩＡＢに出力させる。

【０３２３】Ｔ４のφで、ＤＢＲ２５の内容をＧＢに読み出し、これをＡＬＵ２７に入力する。Ｔ４のφ＃で、ＡＬＵ２７はＧＢから入力した内容をそのまま、ＷＢに出力し、ＥＸＲに格納する。

【０３２４】ステップ２で、リードした結果をＣＣＲ３１に格納する。ＭＡＢ３３に格納した内容を、ＭＡＢ３３でインクリメントさせる。この内容でリードを行う。なお、ＭＡＢ３３のインクリメント機能は、特開平４－３３３１５３号公報などに記載されている。

【０３２５】Ｔ５のφで、ＤＢＲ２５の内容をＧＢに読み出し、これをＡＬＵ２７に入力する。Ｔ５のφ＃で、ＡＬＵ２７はＧＢから入力した内容をそのまま、ＷＢに出力し、ＣＣＲ３１に格納する。

【０３２６】ステップ３で、ＳＰの内容をインクリメント（＋４）する。

【０３２７】Ｔ６のφで、ＳＰの内容を読み出して、ＧＢ経由で、ＡＬＵ２６でインクリメント（＋４）する。

【０３２８】Ｔ６のφ＃で、インクリメントした結果を、ＷＢ経由でＳＰに格納する。

【０３２９】ステップ４で、ＤＢＲ２５に格納した、スタックから復帰したＰＣ３０の内容をアドレスとして、命令のリードを行う。ＤＢＲ２５の内容をインクリメントし、ＰＣ３０に格納する。

【０３３０】Ｔ６のφ＃で、同時に、ＤＢＲ２５に格納したベクタアドレスのリード内容（分岐先の先頭アドレス）をＧＢ経由で、ＭＡＢに転送し、ＩＡＢに出力させ、ＡＬＵでインクリメント（＋２）する。

【０３３１】Ｔ７のφで、インクリメントした結果を、ＷＢ経由でＰＣ３０に格納する。

【０３３２】ステップ５で、ＰＣ３０の内容をアドレスとして、命令のリードを行い、ＰＣ３０のインクリメントを行う。

【０３３３】Ｔ７のφ＃で、ＰＣ３０の内容（分岐先の先頭アドレス）をＧＢ経由で、ＭＡＢ３３に転送し、Ｉ

ＡＢに出力させ、ＡＬＵ２６でインクリメント（＋２）する。リードした命令をＩＲに格納する。

【０３３４】Ｔ８のφで、インクリメントした結果を、ＷＢ経由でＰＣ３０に格納する。次の命令の実行を開始させる。

【０３３５】例えば、ＩＮＴＭ１信号が０レベルの場合には、前記従来ＣＰＵ（例えば、前記平成５年６月（株）日立製作所発行『Ｈ８／３００Ｈシリーズプログラミングマニュアル』に記載のＣＰＵ）と同一のスタックの構造とされる。命令コードが共通であることと相俟って、従来ＣＰＵによって書かれたプログラムをそのまま実行することができる。

【０３３６】新たな、コンディションコードや割込みマスクビットやトレースビットなどを追加する場合には、これに対応したプログラムを作成することになるから、スタックの構造が異なっても実質的な問題はない。割込みマスクビットを追加するなどして、使い勝手を向上することができる。

【０３３７】なお、前記の通りＩＮＴＭ１ビットがＳＹＳＣＲに存在し、このビットの状態がＩＮＴＭ１信号に反映されるようになっている。リセット後に、かかるＳＹＳＣＲの設定を行うことにより、ＥＸＲを使用するかしないかが選択される。

【０３３８】図４４にＣＯＮＴ２２の一部の論理を示す。このＣＯＮＴ２２は、アンド回路８６Ａ～８６Ｄによって構成される。

【０３３９】ＥＸＲを使用しない、すなわち、ＩＮＴＭ１ビットを”０”にクリアすると、ＥＸＲのビットは全て”０”とみなされ、設定値は、無視されるようにされる。

【０３４０】次に、図４におけるエミュレーション用インタフェース３９に含まれる制御レジスタ４１の構成を示す。この制御レジスタ４１は、以下説明するように、（１）ＡＳＥコントロールレジスタ　Ｄ　（ＡＳＥＣＲＤ）、（２）ブレークコントロールレジスタ　Ａ　Ｂ（ＢＲＣＲＡ、Ｂ）、（３）ブレークアドレスレジスタＡ、Ｂ（ＢＡＲＡ、Ｂ）、（４）ブレークアドレスマスクレジスタ　Ａ、Ｂ（ＢＡＭＲＡ、Ｂ）、および（５）ＡＳＥ専用スタックレジスタ（ＢＲＫＳＴＫＲ）から構成されている。

【０３４１】図４５に、（１）ＡＳＥコントロールレジスタ　Ｄ　（ＡＳＥＣＲＤ）の構成を示す。このレジスタは８ビットリード／ライト可能なレジスタで、シングルステップの設定、ＲＴＢ命令実行後の割込制御、多重ブレークの許可禁止、ウインドウ機能を指定する。各ビットの内容を表２１乃至表２４に示す。

【０３４２】

【表２１】

表２１

ビット7：シングルステップイネーブル（SSTPE）

シングルステップを設定します。

ビット7 SSTPE	説明
0	シングルステップ解除（初期値）
1	シングルステップ設定

【０３４３】

【表２２】

表２２

ビット6：RTB割込イネーブル（RTBINTE）

RTB命令実効終了後の割込受付の許可／禁止を選択します。

ビット6 RTBINTE	説明
0	RTB命令実効終了後の割込を禁止（必ず1命令実行します）
1	RTB命令実効終了後の割込を許可（初期値）

【０３４４】

【表２３】

表２３

ビット5：多重ブレークイネーブル（MBIE）

ブレークモード中のブレーク割込（ASEBRK#端子によるブレーク割込）の許可／禁止を選択します。

ビット5	説明
MBIE	
0	ブレークモード中のブレーク割込を禁止（初期値）
1	ブレークモード中のブレーク割込を許可

【０３４５】

【表２４】

表２４

ビット1、0：ウィンドウセレクト1、0（WD1、WD0）

ブレークモード時のウィンドウ機能を設定します。これによって、ブレークモード時のユーザ空間へのCPUのデータアクセスを許可することができます。

ビット1	ビット0	説明
WD1	WD0	
0	0	ウィンドウ機能を解除（初期値）
0	1	CPUデータリードサイクルに対し、ウィンドウを設定
1	0	CPUデータライトサイクルに対し、ウィンドウを設定
1	1	CPUデータリード／ライトサイクルに対し、ウィンドウを設定

【０３４６】図４６に、（２）ブレークコントロールレジスタ　Ａ　Ｂ（ＢＲＣＲＡ、Ｂ）の構成を示す。このレジスタは、（ａ）ＢＲＣＲＡ、（ｂ）ＢＲＣＲＢからなり、各々は８ビットのリード／ライトが可能なレジスタで、それぞれＰＣブレークのチャネルＡ、Ｂの制御を行う。各ビットの内容を表２５乃至表２８に示す。

【０３４７】

【表２５】

表２５

ビット7：コンディションマッチフラグA/B（CMFA/B）

チャネルA/Bの設定した条件が成立したことを示します。

ビット7	説明
CMFA/B	
0	[クリア条件] 0をライトしたとき（初期値）
1	[セット条件] チャネルA/Bの設定した条件が成立したとき

【０３４８】

【表２６】

表２６

ビット5：トリガアウトプットイネーブルA/B（TOEA/B）

チャネルA/Bのトリガ出力信号（ASEBKTOA/B#）の出力の許可／禁止を選択します。

ビット6	説明
TOEA/B	
0	トリガ出力（ASEBKTOA/B#）は禁止（ハイレベル固定）（初期値）
1	トリガ出力（ASEBKTOA/B#）は許可（条件成立時1ステートロウレベル出力）

【０３４９】

【表２７】

73　　74

表２７

ビット2、1：コンディションセレクトA/B1、0（CSELA/B1、CSELA/B0）

チャネルA/Bのブレーク条件を選択します。指定した条件が成立し、アドレス比較が一致すると、PCブレークが要求されます。対象はCPUのみです。

ビット2	ビット1	説明
CSELA/B1	CSELA/B0	
0	0	命令実行（初期値）
0	1	CPUデータリードサイクル
1	0	CPUデータライトサイクル
1	1	CPUデータリード／ライトサイクル

【０３５０】

【表２８】

表２８

ビット0：ブレーク割込イネーブルA/B（BIEA/B）

チャネルA/BのPCブレークの許可／禁止を選択します。

ビット0	説明
BIEA/B	
0	チャネルA/BのPCブレーク禁止（初期値）
1	チャネルA/BのPCブレーク許可

【０３５１】図４７に、（３）ブレークアドレスレジスタ　A、B（BARA、B）の構成を示す。このレジスタは、（a）BARA、（b）BARBからなり、各々は３２ビットのリード／ライトが可能なレジスタで、それぞれＰＣブレークのチャネルA、Bのアドレスを指定する。３２ビットのレジスタをバイトサイズに分割して、BARR、E、H、Lと表記される場合もある。最上位のＢＡＲＲはリザーブされている。リードすると不定値が読み出される。ライトは無効である。

【０３５２】図４８に、（４）ブレークアドレスマスクレジスタ　A、B（BAMRA、B）の構成を示す。このレジスタは、（a）BAMRA、（b）BAMRBからなり、各々は３２ビットのリード／ライトが可能なレジスタで、それぞれＰＣブレークのチャネルA、Bのアドレス比較のマスクを行うビットを指定する。BAMRのビットを”１”にセットすると、このビットに対応するアドレスのビットは、アドレス比較対象から除外される。３２ビットのレジスタをバイトサイズに分割して、BAMRR、E、H、Lと表記される場合もある。最上位のＢＡＲＲはリザーブされている。リードすると不定値が読み出される。ライトは無効である。

【０３５３】図４９に、（５）ＡＳＥ専用スタックレジスタ（BRKSTKR）の構成を示す。このレジスタは、６バイト（４８ビット）のリード／ライトが可能なレジスタで、ユーザモード⇔ブレークモードの遷移時に、スタック領域として使用する。ユーザのＳＰは使用せず、保持される。スタックされるリソースおよびスタックの構造は、ＭＣＵ動作モード（ノーマルモード／アドバンストモード）および、制御レジスタの設定（ＳＹＳＣＲのＩＮＴＭ１ビット）によって相違される。表２９にこのレジスタの使用方法を示す。

【０３５４】

【表２９】

表２９

ASE専用スタックレジスタの使用方法

相対アドレス	ノーマルモード		アドバンストモード	
	INTM1=0	INTM1=1	INTM1=0	INTM1=1
0		EXR		EXR
1		リザーブ		リザーブ
2	CCR	CCR	CCR	CCR
3	CCR	CCR	PC	PC
4	PC	PC		
5				

【０３５５】エミュレーション用ソフトウェアの実行状態への遷移（ブレーク）時には、固定アドレスのブレークスタックレジスタを使用するようにする。ブレーク例外処理や、ブレークからのリターン命令時には、ユーザのスタックポインタ（ＥＲ７）を使用せず、固定的なスタックアドレスを生成する。かかるスタックアドレスの生成はＥＭＬＳＰ２９による。

【０３５６】図５０に、ＥＭＬＳＰ２９の構成を示す。このＥＭＬＳＰ２９は、クロックトバッファで構成される。

【０３５７】かかるクロックトバッファの内、ビット２３～１０は１固定、ビット５、４、０は０固定、ビット９～６は、外部からの指定を入力する。また、ビット３、２、１はＣＯＮＴ２２の制御信号を入力する。ＣＭＯＳ回路で構成する場合、必要に応じて論理反転を用いればよい。クロックトバッファの出力はＧＢに接続されている。また制御信号ｍは、ＣＯＮＴの制御信号とクロック（φ＃）の論理積信号である。

【０３５８】通常のレジスタ回路が、データを保持するためのラッチ回路を持たなければならないが、ＥＭＬＳＰ２９は、これを持たず、小型化を図っている。

【０３５９】従って、ブレークスタックレジスタの先頭アドレスは、Ｂ’０００００００であって、６４ｋバイト単位で１６通りのアドレスを選択可能とされる。マイクロコンピュータの内部Ｉ／Ｏレジスタの配置によって、アドレスを変更できる。

【０３６０】ブレーク例外処理の実行シーケンスは、図３７、３８と同様であり、そこでのＳＰ（ＥＲ７）の読み出しに代わって、ＥＭＬＳＰ２９を読み出すようにする。

【０３６１】この場合、最初（Ｔ３のφ）は下位アドレスをＢ’０００１１０として、デクリメントした内容のＢ’０００１００がスタックのアドレスとされる。

【０３６２】２回目（Ｔ４のφ）は下位アドレスをＢ’０００１００として、デクリメントした内容のＢ’０００００１０がスタックのアドレスとされる。

【０３６３】３回目（Ｔ５のφ）は下位アドレスをＢ’０００００１０として、デクリメントした内容のＢ’０００００００がスタックのアドレスとされる。３回目は、ＩＮＴＭ信号が活性状態のときに有効である。読み出されるビット２、１は、ＣＯＮＴ２２の制御信号によって選択する。

【０３６４】リターン命令の実行シーケンスは、図４２と同様であり、ＳＰ（ＥＲ７）の読み出しに代わって、ＥＭＬＳＰ２９を読み出すようにする。

【０３６５】最初（Ｔ２のφ＃）はＩＮＴＭ１信号によって異なり、ＩＮＴＭ１信号が非活性状態であれば、下位アドレスをＢ’０００００１０として、ＩＮＴＭ１信号が活性状態であれば、下位アドレスをＢ’０００００００として、読み出す。これらがスタックのアドレスとなる。

【０３６６】２回目（Ｔ３のφ）は、ＩＮＴＭ１信号が活性状態である場合に有効であり、下位アドレスをＢ’０００００１０として、デクリメントした結果をアドレスとしてリードを行う。

【０３６７】３回目はＭＡＢ３３のインクリメントによってアドレスを生成し、ＥＭＬＳＰ２９は使用しない。

【０３６８】これにより、固定的な出力回路として論理規模を縮小できる。ユーザに公開されない資源による論理規模の増大を最小限にすることができる。

【０３６９】図５１に、ブレーク例外処理の実行タイミングを示す。実行シーケンスは図３７、３８の例外処理タイミングと同様である。

【０３７０】前記同様に、Ｔ２から割り込み例外処理の実行が開始される。プリフェッチした命令はキャンセルされ、図示されないブレーク要求信号に呼応して、ＣＯＮＴ２２の入力が切り替えられる。

【０３７１】Ｔ３のバス動作が行われない期間に、ブレークモードを示す信号ＢＲＫＡＫ＃が活性状態になる。

【０３７２】図５２に、ブレーク制御論理の回路構成を示す。この回路は、アンド回路８１Ａ乃至８１Ｇ、オア回路８２Ａ乃至８２Ｄ、フリップフロップ８３から構成されている。

【０３７３】ＣＰＵに対するブレーク要求は、３要因が

存在する。第１はＢＲＫ端子による要求である。第２はアドレス比較Ａによる要求であり、これは、ＢＲＫＣＲのＢＩＥＡビットによって許可される。第３はアドレス比較Ｂによる要求であり、これは、ＢＲＫＣＲのＢＩＥＢビットによって許可される。なお、かかるアドレス比較は、前記の通り、（ＣＡ２３・ＡＲ２３＋￢ＣＡ２３・￢ＡＲ２３＋ＡＭＲ２３）・…・（ＣＡｎ・ＡＲｎ＋￢ＣＡｎ・￢ＡＲｎ＋ＡＭＲｎ）・…・（ＣＡ０・ＡＲ０＋￢ＣＡ０・￢ＡＲ０＋ＡＭＲ０）と表現される。（￢は論理反転を示す）。

【０３７４】これらの論理和信号が、ブレーク要求として、ＣＰＵに与えられる。

【０３７５】ブレークモードでは、ＭＢＩＥビットの状態によってＢＲＫ端子によるブレーク要求の許可禁止が選択される。即ち、ブレークモードでＭＢＩＥビットが”０”にクリアされている場合は、ブレーク要求が抑止される。アドレス比較によるブレーク要求は、ブレークモードで禁止される。

【０３７６】また、ＭＢＩＥビットは、フリップフロップで構成され、ＢＲＫＡＫ信号の反転信号で”０”にクリアされる。かかるフリップフロップの入力は、所定のデータバスのビットであって、クロックは、ブレークモード信号とアドレスデコード信号とライト信号の論理積信号とされる。かかるアドレスデコード信号は、ＣＰＵの出力するアドレスがＢＲＫＣＲの存在するアドレスになったとき、活性状態とされる。即ち、ブレークモードでのみライト可能とされる。

【０３７７】即ち、ブレークモードに遷移した直後は、ブレーク要求がが禁止状態であって、不所望のブレークの多重例外処理（スタックした内容の破壊）が禁止される。また、ＢＲＫＣＲのＳＳＴＰビットと、ＢＲＫＡＫ信号の反転信号との論理積が、シングルステップブレーク要求として、ＣＰＵに与えられる。

【０３７８】シングルステップブレーク要求と、ＲＴＢ命令実行信号との論理積信号と、ブレーク要求が、ＣＰＵ内部で、ＣＰＵブレーク例外処理要求として認識される。これらの例外処理の内容は共通とされる。

【０３７９】ＣＰＵはＲＴＢ命令実行時には、かかるシングルステップブレーク要求を無視する。

【０３８０】上記実施例によれば、以下の作用効果を得るものである。

【０３８１】（１）既存の命令セットと互換性を維持しつつ、乗算器を内蔵することに当っては、ポストインクリメントレジスタ間接のアドレッシングモードのみをサポートすることによって、アドレッシングモードの増加を最小限にして、かつ処理性能を低下させずに積和演算を実行可能にすることができる。また、アドレスレジスタの補正をレジスタ間演算命令で行い、これを１ステートで実行することができ、アドレッシングの柔軟性を向上することができる。さらに、積和演算をＣＰＵの内部動作（ポストインクリメントのアドレス計算）と並行に行うことによって、実行ステート数の短縮を行うことができる。

【０３８２】（２）乗算器を利用して乗算命令を実行することにあたっては、乗算の結果（積、フラグ）を直接汎用レジスタ、ＣＣＲに格納するようにして、直ちに結果を利用できるようにし、実質的な乗算の実行速度を向上することができる。また、積和演算の結果（ＭＡＣ）をリード（ＳＴＭＡＣ）すると同時に、乗算器内部で保持したフラグをＣＣＲに格納することによって、積和演算結果の利用や判定を容易に行うことができ、使い勝手を向上することができる。

【０３８３】（３）乗算器を取外し可能にすることによって、乗算器を取外した場合は、積和演算をサポートしないことによって、容易に下位ＣＰＵを実現し、論理的・物理的規模を縮小し、製造費用を低減した別のマイクロコンピュータを容易に開発することができる。また、汎用的な乗算命令を、乗算器によらずにサポートすることによって、かかる別のマイクロコンピュータにおける使い勝手の低下を防止できる。さらに、乗算器によらない乗算命令を除算と同一のシーケンスで実行するようにして、乗算器を持つマイクロコンピュータにおいても冗長な論理を最低限にすることができる。

【０３８４】また、乗算器使用するか使用しないかの制御信号を与えて制御することによって、テスト性を向上したり、エミュレータを共通化したりすることができる。全体的な開発効率を向上することができる。さらにまた、乗算器を削除し、小型化したＣＰＵを用いて、マイクロコンピュータを構成することによって、半導体集積回路の論理規模・物理的規模を縮小して、製造費用の縮小を図ることができる。

【０３８５】（４）乗算器とＣＰＵを一体に構成して、乗算器・ＣＰＵ間の配線を短縮して、物理的規模を縮小する。また、高速化に寄与することができる。

【０３８６】（５）乗算器のテストモードを設定して、このときの乗算器の処理を１ステップのみにすることによって、論理規模の増加を最低限にして、テストの容易性を向上することができる。テストステップを短縮することができる。

【０３８７】（６）内部動作のパイプラインに対応して、入出力タイミングの異なるレジスタ選択回路を複数持つことにより、実質的に１命令／１ステート実行を行うことができる。

【０３８８】（７）複数レジスタの退避／復帰命令を持ち、この組み合せを固定的にすることによって、論理規模の縮小を図ることができる。レジスタの本数の異なる命令を複数命令サポートすることによって、使い勝手の低下を防ぐことができる。また、複数レジスタの退避／復帰命令を関数（サブルーチン）の入り口／出口で実行することによって、Ｃ言語などで記述された場合のよう

に、関数の使用頻度が高い場合に、処理速度を特に向上することができる。さらに、割り込み例外処理ルーチンの先頭で複数レジスタの退避を用いることにより、リアルタイム性の向上を図ることができる。

【０３８９】（８）ＥＸＲの有効／無効を切り換えることで、互換性を維持することと、機能拡張とを両立することができる。ＥＸＲを無効とし、例外処理において退避／復帰を行わないようにすることで、スタックの節約と、割込み応答時間の高速化に寄与することができる。また、互換性を維持する。さらに、ＥＸＲを有効とすることで、割り込みマスクレベルを拡張したり、トレース機能を追加したりして、使い勝手を向上することができる。

【０３９０】（９）エミュレータ用の固定的なスタックポインタを持つことによって、ユーザプログラムとエミュレーションプログラムの遷移時に、ユーザーのスタックポインタとは独立して、固定的なアドレスに対して退避および復帰が行われるから、エミュレータのソフトウェア、ハードウェアの開発を容易にすることができる。

【０３９１】また、エミュレータ用のスタックポインタを固定的にすることによって、ユーザに公開しない資源を最小限の論理的・物理的規模にすることができる。ユーザプログラムからエミュレーションプログラムへの遷移（ブレーク）を多重に行うことを禁止することを可能にすることによって、不所望のスタックの内容の破壊を防止することができる。

【０３９２】以上本発明者等によってなされた発明を実施例に限定されるものではなく、その要旨を逸脱しない範囲において種々変更可能である。実施例を相互に組み合せて使用することもできる。

【０３９３】例えば、ＣＰＵの命令セットやレジスタ構成は変更可能である。内部バス幅なども変更可能である。但し、命令の大部分の命令コード長より、小さいバス幅でないことが望ましい。

【０３９４】また、乗算器の内部構成なども種々変更可能である。１６ビット×６ビットを３回繰り返すのではなく、１６ビット×４ビットを４下位繰り返すようにしてもよい。命令実行ステートと論理的規模に鑑みて選択すればよい。

【０３９５】さらに、飽和演算の指定、乗算器あり／なしの指定、テストモードの指定方法が種々変更可能であることは言うまでもない。

【０３９６】さらにまた、乗算器に限らず除算器を内蔵するものであっても良い。

【０３９７】また、互換性を維持すべき対象は、前記例に限定されない。一般的に、ＣＰＵの例外処理時に、コントロールレジスタの内容を待避することは行われており、そのほかのＣＰＵについても、本発明を適用して、互換性を維持しつつ、コントロールレジスタの機能を拡張することができる。

【０３９８】さらに、シングルチップマイクロコンピュータのその他の機能ブロックについても何等制約されない。

【０３９９】以上の説明では主として本発明者によってなされた発明をその背景となった利用分野であるシングルチップマイクロコンピュータに適用した場合について説明したが、それに限定されるものではなく、その他のデータ処理装置にも適用可能であり、本発明は少なくとも、複数の動作モードを選択して動作するデータ処理装置に適用することができる。

【０４００】

【発明の効果】本題において開示される発明のうち代表的なものによって得られる効果を簡単に説明すれば下記の通りである。

【０４０１】（１）乗算器をＣＰＵに内蔵することによって、アドレッシングモードの増加を最小限にして、かつ処理性能を低下させずに積和演算を実行可能にすることができる。乗算器による乗算の結果を汎用レジスタＣＣＲに反映させ、かかる結果を直ちに利用可能にして、処理速度を高速にすることができる。

【０４０２】（２）乗算器を取外し可能に（独立して）設けることによって、乗算器を取外した場合は、積和演算をサポートしないことによって、容易に下位ＣＰＵを実現し、論理的・物理的規模を縮小し、製造費用の低減に寄与することができる。

【０４０３】（３）複数レジスタの退避／復帰命令を持ち、この組み合せを固定的にすることによって、論理規模の縮小を図ることができ、また、高速化を図ることができる。

【０４０４】（４）コントロールレジスタの有効／無効を切り換えることで、スタックの節約と、割込み応答時間の高速化に寄与することができるとともに、互換性を維持することができる。

【０４０５】（５）エミュレータ専用の固定スタックポインタを持つことにより、エミュレータをサポートすることができ、また、論理規模の増加を最低限にして、エミュレータの設計を容易にすることができ、さらにエミュレータ専用スタックポインタの一部のアドレスを、ＣＰＵ外部から与えるようにして、スタックレジスタをリロケータブルにし、マイクロコンピュータのアドレス配置などに容易に対応することができる。

【図面の簡単な説明】

【図１】本発明の実施例によるマイクロコンピュータの構成を示すブロック図である。

【図２】本実施例のマイクロコンピュータのシステムコントロールレジスタの構成図である

【図３】本実施例のマイクロコンピュータに用いられるＣＰＵの命令フォーマットの構成図である。

【図４】本実施例のマイクロコンピュータに用いられるＣＰＵと乗算器を示すブロック図である。

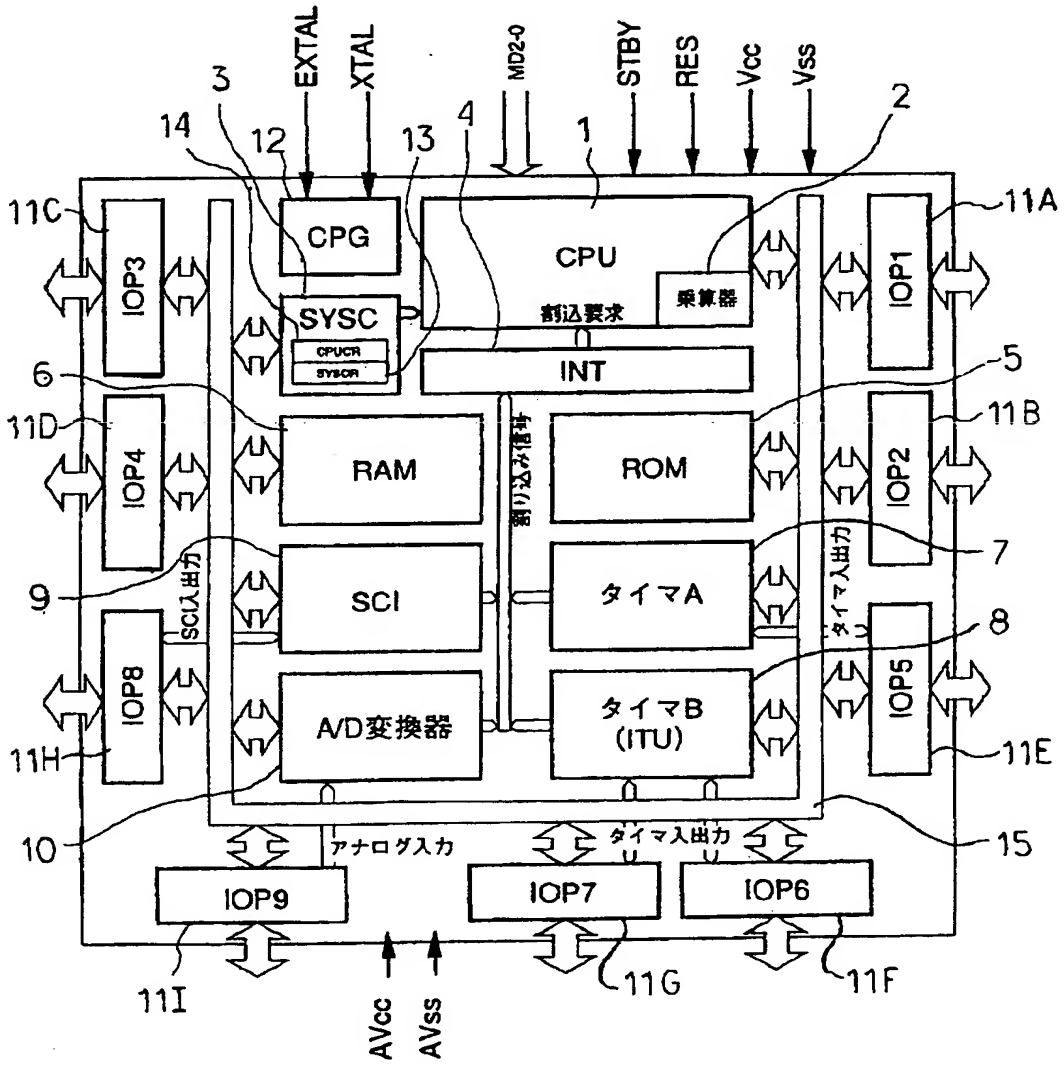
【図５】本実施例のマイクロコンピュータにおける制御レジスタの１ビットの構成図である。
【図６】本実施例のマイクロコンピュータにおける制御信号の設定方法の一例を示すブロックである。
【図７】本実施例のマイクロコンピュータにおける制御信号の設定方法の一例を示す概略図である。
【図８】本実施例のマイクロコンピュータに用いられるＣＰＵのレジスタの構成図である。
【図９】本実施例のマイクロコンピュータに用いられるＣＰＵのレジスタの構成図である。
【図１０】本実施例のマイクロコンピュータに用いられるＣＰＵのレジスタの使用方法を説明するブロック図である。
【図１１】本実施例のマイクロコンピュータに用いられるＣＰＵのレジスタのスタックの状態の説明図である。
【図１２】本実施例のマイクロコンピュータに用いられるＣＰＵの基本動作のタイミング図である。
【図１３】本実施例のマイクロコンピュータに用いられるＣＰＵの基本動作のタイミング図である。
【図１４】本実施例のマイクロコンピュータに用いられる乗算器の構成を示すブロック図である。
【図１５】本実施例のマイクロコンピュータに用いられる乗算器による演算方法の説明図である。
【図１６】本実施例のマイクロコンピュータに用いられる乗算器のフラグの実現方法の説明図である。
【図１７】本実施例のマイクロコンピュータに用いられる乗算器のフラグの実現方法の説明図である。
【図１８】本実施例のマイクロコンピュータに用いられるバススイッチの構成を示すブロック図である。
【図１９】本実施例のマイクロコンピュータにおけるＭＡＣ命令の動作のタイミング図である。
【図２０】図１９に連続する動作のタイミング図である。
【図２１】本実施例のマイクロコンピュータにおけるＳＴＭＡＣ命令およびＬＤＭＡＣ命令の動作のタイミング図である。
【図２２】図２１に連続する動作のタイミング図である。
【図２３】本実施例のマイクロコンピュータにおいて乗算器を用いた場合の乗算命令の動作のタイミング図である。
【図２４】図１９に連続する動作のタイミング図である。
【図２５】本実施例のマイクロコンピュータにおいて乗算器を用いない場合の乗算命令の動作のタイミング図である。
【図２６】図２５に連続する動作のタイミング図である。
【図２７】本実施例のマイクロコンピュータにおける乗算命令の状態遷移図である。
【図２８】本実施例のマイクロコンピュータに用いられる演算器の構成を示すブロック図である。
【図２９】本実施例のマイクロコンピュータにおける複数レジスタの退避命令の実行シーケンス図である。
【図３０】図２９に連続する動作のタイミング図である。
【図３１】本実施例のマイクロコンピュータにおける複数レジスタの復帰命令の実行シーケンス図である。
【図３２】図３１に連続する動作のタイミング図である。
【図３３】本実施例のマイクロコンピュータにおけるＲＳＥＬ２入力制御回路の構成を示すブロック図である。
【図３４】図３３の動作の説明図である。
【図３５】本実施例のマイクロコンピュータに適用されるＣ言語による変換リストの概略例である。
【図３６】本実施例のマイクロコンピュータに適用されるＣ言語による変換リストの概略例である。
【図３７】本実施例のマイクロコンピュータにおける割込例外処理の実行シーケンス図である。
【図３８】図３７に連続する動作のタイミング図である。
【図３９】本実施例のマイクロコンピュータにおける例外処理の状態遷移図である。
【図４０】本実施例のマイクロコンピュータにおける例外処理後のスタックの状態の説明図である。
【図４１】本実施例のマイクロコンピュータにおける例外処理後のスタックの状態の説明図である。
【図４２】本実施例のマイクロコンピュータにおけるＲＴＥ命令の実行シーケンス図である。
【図４３】本実施例のマイクロコンピュータにおける例外処理の状態遷移図である。
【図４４】本実施例のマイクロコンピュータにおける制御回路の構成の説明図である。
【図４５】本実施例のマイクロコンピュータにおける制御レジスタの構成図である。
【図４６】本実施例のマイクロコンピュータにおける制御レジスタの構成図である。
【図４７】本実施例のマイクロコンピュータにおける制御レジスタの構成図である。
【図４８】本実施例のマイクロコンピュータにおける制御レジスタの構成図である。
【図４９】本実施例のマイクロコンピュータにおける制御レジスタの構成図である。
【図５０】本実施例のマイクロコンピュータにおけるエミュレーションスタックポインタの構成の説明図である。
【図５１】本実施例のマイクロコンピュータにおけるブレーク割込みシーケンスの実行タイミング図である。
【図５２】本実施例のマイクロコンピュータにおけるブレーク制御処理の回路構成図である。

83　　84

【符号の説明】
１…ＣＰＵ、２…乗算器、３…システムコントローラ（ＳＹＳＣ）、４…割込コントローラ（ＩＮＴ）、５…ＲＯＭ、６…ＲＡＭ、９…シリアルコミュニケーションインターフェース（ＳＣＩ）、１３…システムコントロールレジスタ（ＳＹＳＣＲ）、１４…制御レジスタ（ＣＰＵＣＲ）、２１…命令レジスタ（ＩＲ）、２２…命令デコーダ・制御回路（ＣＯＮＴ）、２３…レジスタセレクタ（ＲＳＥＬ）、２４…ライトデータバッファ（ＤＢＷ）、２５…リードデータバッファ（ＤＢＲ）、２６、２７…演算器、２９…エミュレータスタックポインタ（ＥＭＬＳＰ）、３０…プログラムカウンタ（ＰＣ）、３１…コンディションレジスタ（ＣＣＲ）、３２…拡張レジスタ（ＥＸＲ）、３３…アドレスバッファ（ＭＡＢ）、３４…バススイッチ、３８…エミュレーション用プロセッサ、３９…エミュレーション用インタフェース、４４…インタフェースケーブル、４８…エミュレーションメモリ、４９…ブレーク制御回路、５０…リアルタイムトレース回路、５８…ＣＭＯＳインバータ回路、６５Ａ～６５Ｃ…デコーダ、６６Ａ～６６Ｃ、７１Ａ、７１Ｂ、７７…選択回路、６７…加算器、７２Ａ～７２Ｃ…拡張回路、７６…算術論理演算回路、７８…シフト回路、７９…制御回路、７５Ａ、７５Ｂ、８１Ａ～８１Ｇ、８６Ａ～８６Ｄ…アンド回路、８０Ａ、８０Ｂ、８２Ａ～８２Ｄ…オア回路、８３…フリップフロップ。

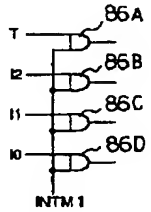
【図１】

図 1



【図４４】

図４４



【図２】

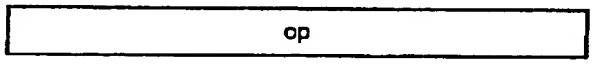
図２

ビット：	7	6	5	4	3	2	1	0
	MACS	-	INTM1	INTM0	NMIEG	-	-	RAME
初期値：	0	0	0	0	0	0	0	1
R/W：	R/W	-	R/W	R/W	R/W	-	-	R/W

【図３】

図３

(1)　オペレーションフィールドのみ

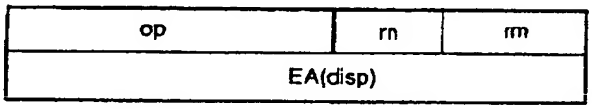


NOP、RTSなど

(2)　オペレーションフィールドとレジスタフィールド

ADD.B　Rn,Rmなど

(3)　オペレーションフィールド、レジスタフィールドおよびEA拡張部



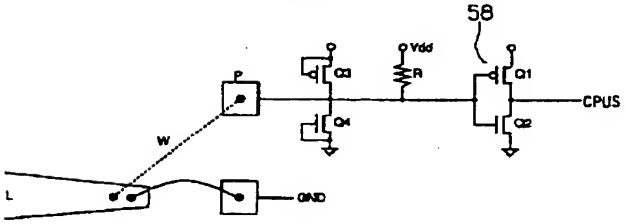
MOV.B　@(d:16,Rn),Rmなど

(4)　オペレーションフィールド、EA拡張部およびコンディションフィールド

BRA　d:16など

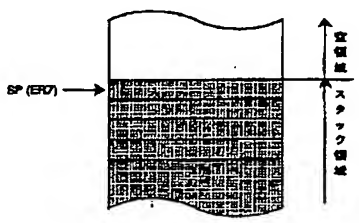
【図７】

図７



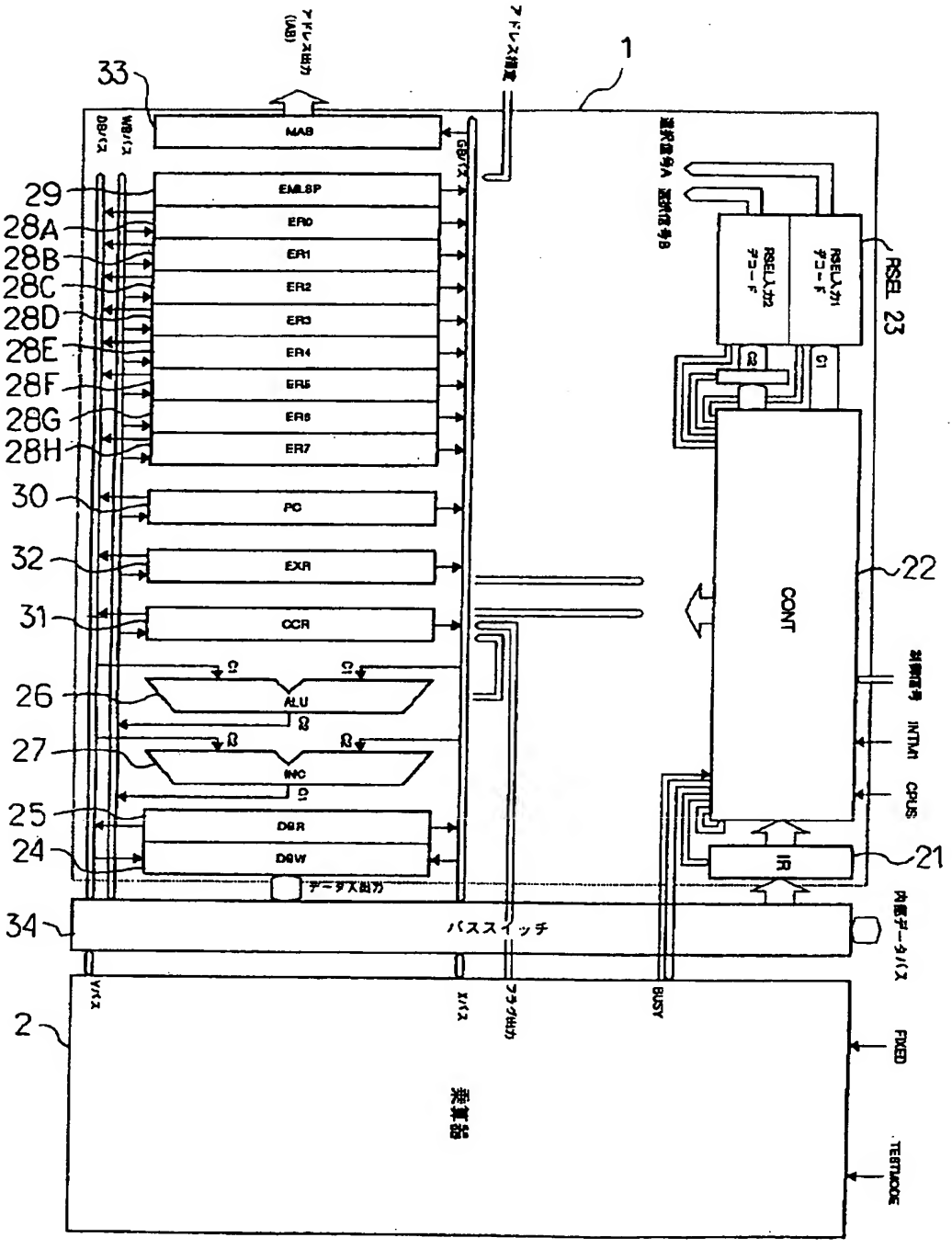
【図１１】

図１１



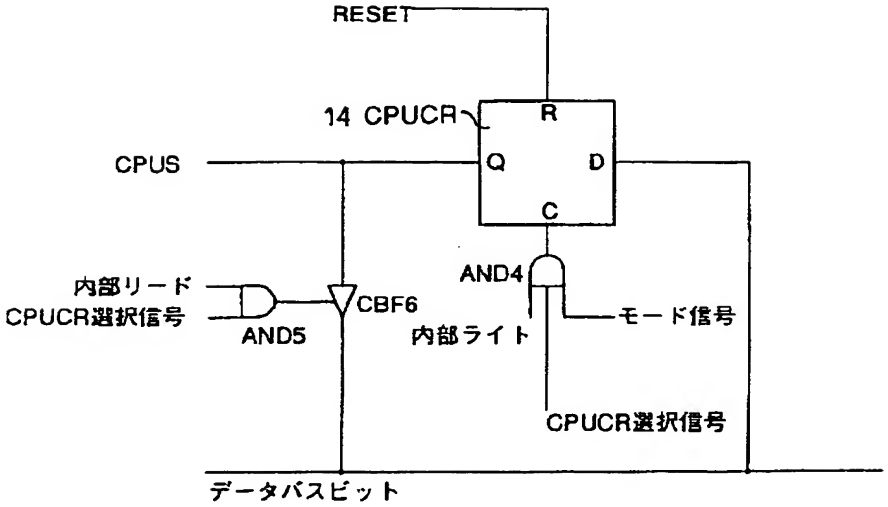
【図４】

図４


33
1
MAB
29
EMLSP
ER0
ER1
ER2
ER3
ER4
ER5
ER6
ER7
28A
28B
28C
28D
28E
28F
28G
28H
30
PC
32
EXR
31
CCR
26
ALU
27
INC
25
DBR
24
DBW
34
バススイッチ
2
演算部
RSEL 23
CONT
22
IR
21
INTM1
CPUS
FIXED
TESTMODE
BUSY

【図５】

図５



【図８】

図８

汎用レジスタ（Rn）

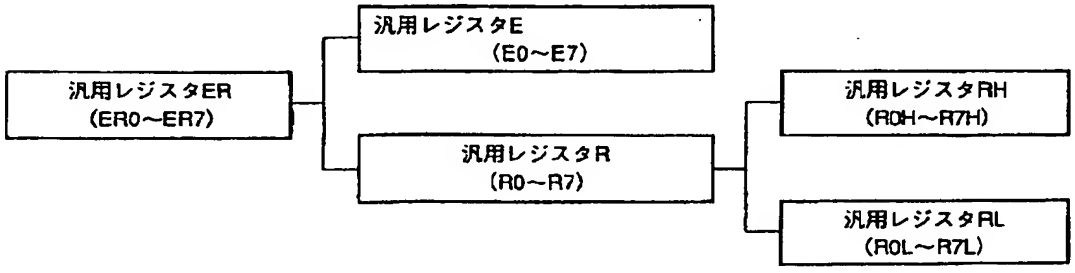
	15	0 7	0 7	0
ER0	E0	R0H	R0L	
ER1	E1	R1H	R1L	
ER2	E2	R2H	R2L	
ER3	E3	R3H	R3L	
ER4	E4	R4H	R4L	
ER5	E5	R5H	R5L	
ER6	E6	R6H	R6L	
ER7(SP)	E7	R7H	R7L	

【図１０】

図１０

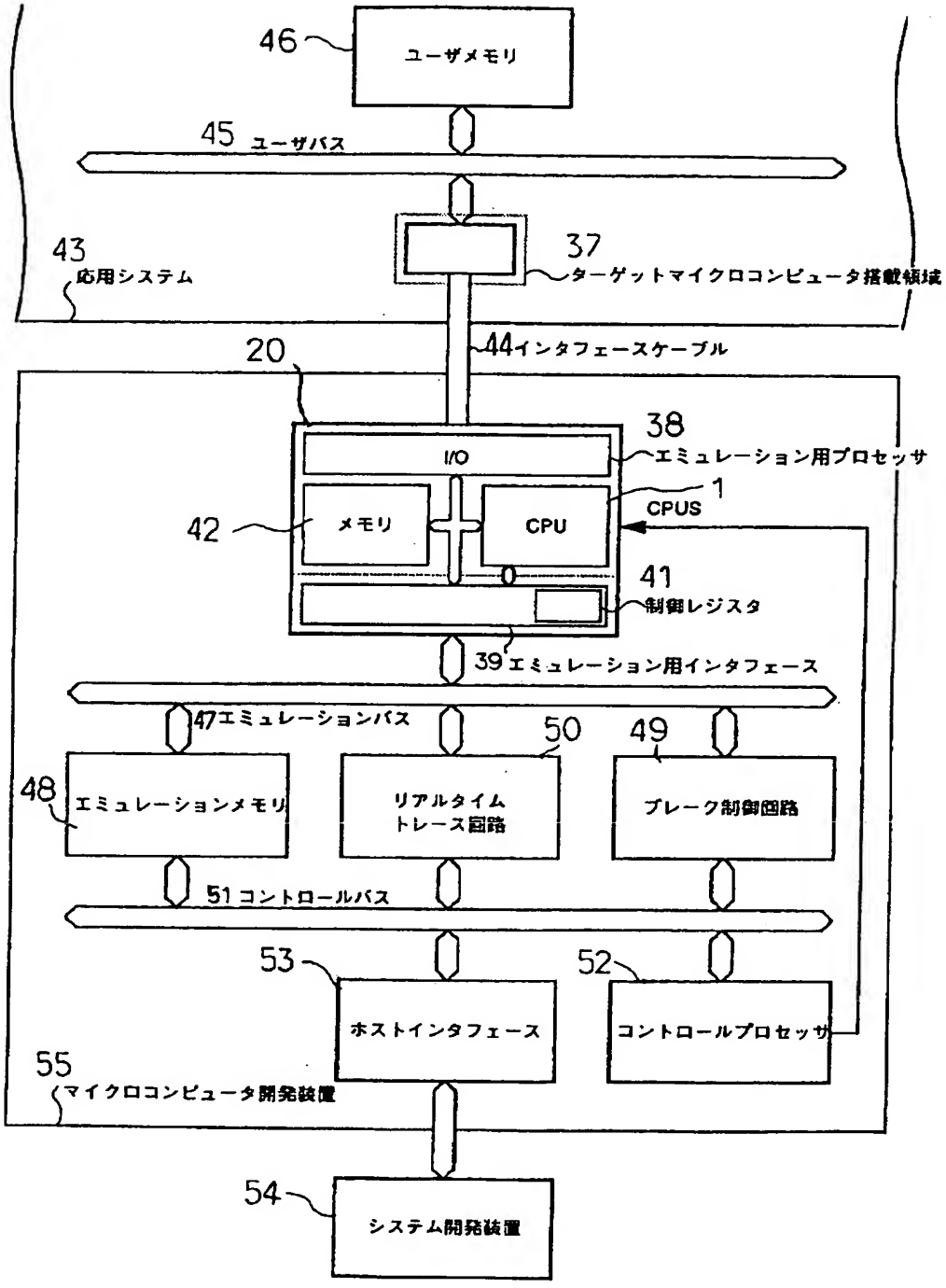
●アドレスレジスタ

●32ビットレジスタ　　●16ビットレジスタ　　●8ビットレジスタ



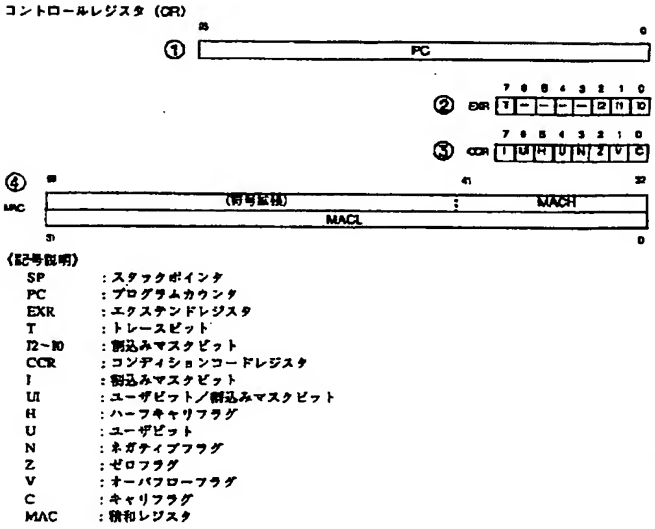
【図６】

図６


46
ユーザメモリ
45 ユーザバス
43
応用システム
37
ターゲットマイクロコンピュータ搭載領域
20
44インタフェースケーブル
38
エミュレーション用プロセッサ
I/O
1
CPUS
42
メモリ
CPU
41
制御レジスタ
39エミュレーション用インタフェース
47エミュレーションバス
50
49
48
エミュレーションメモリ
リアルタイム
トレース回路
ブレーク制御回路
51 コントロールバス
53
52
ホストインタフェース
コントロールプロセッサ
55
マイクロコンピュータ開発装置
54
システム開発装置

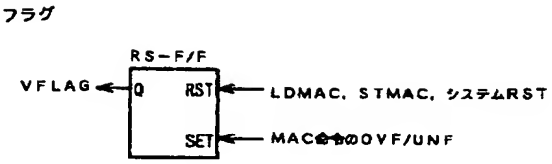
【図９】

図９



【図１６】

図１６



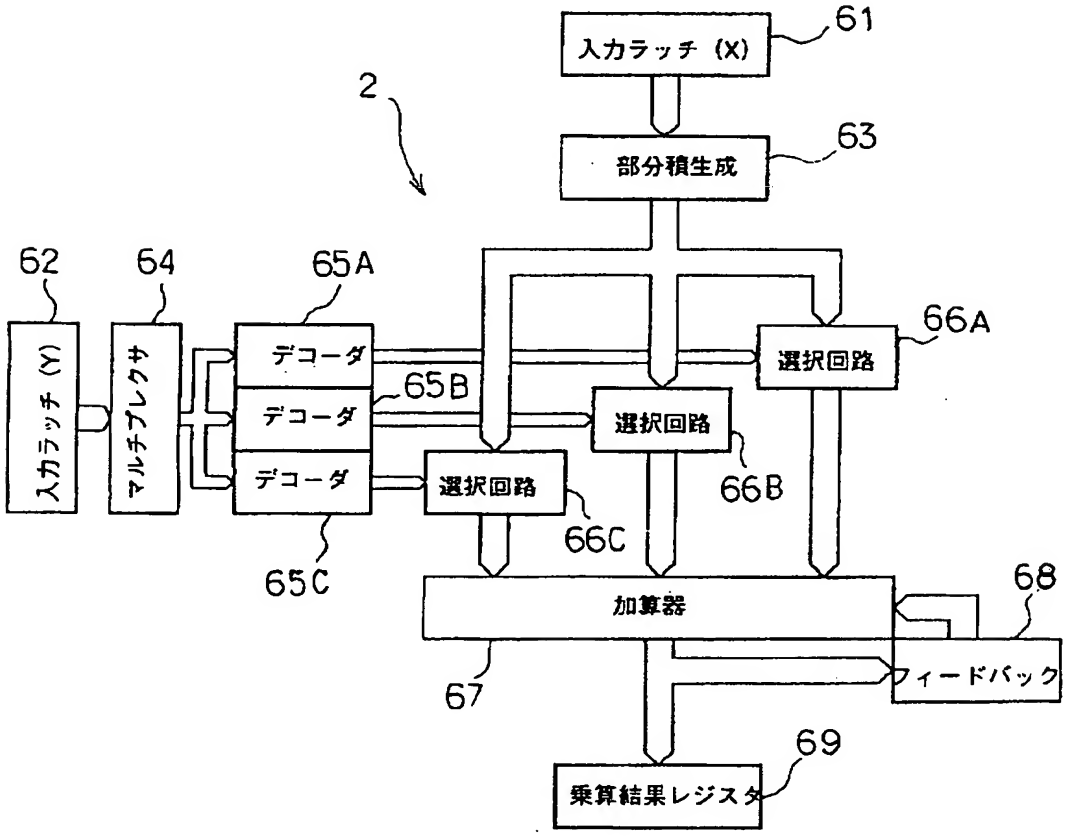
【図４５】

図４５

ビット：	7	6	5	4	3	2	1	0
	SSTPE	RTBINTE	MBIE	-	-	-	WDI	WDO
初期値：	不定/0	不定/1	不定/0	不定	不定	不定	0/0	0/0
R/W：	(R/W)	(R/W)	(R/W)	-	-	-	(R/W)	(R/W)

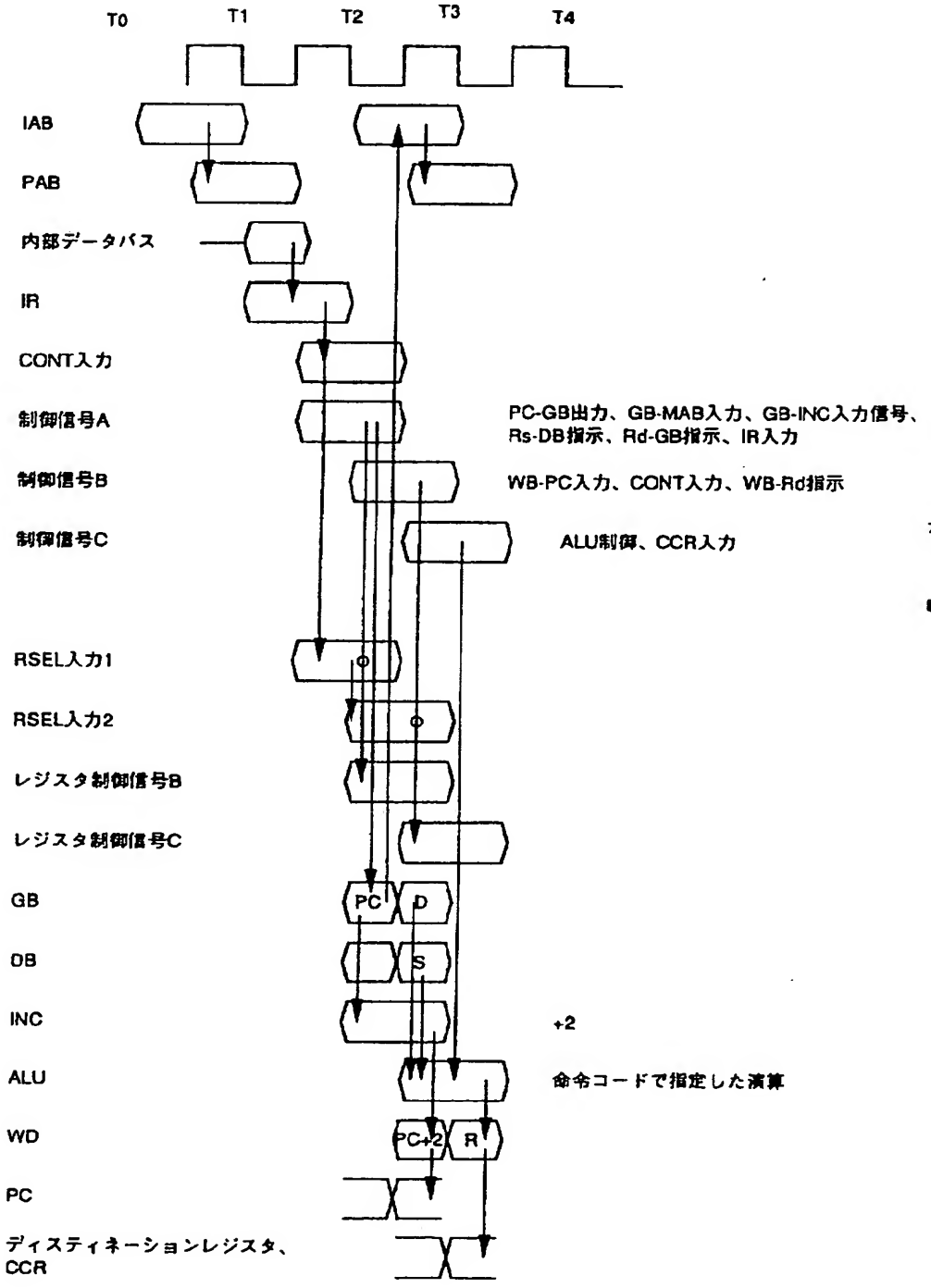
【図１４】

図１４



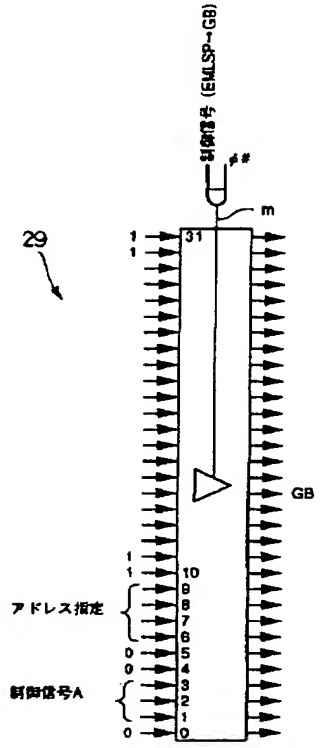
【図１２】

図１２



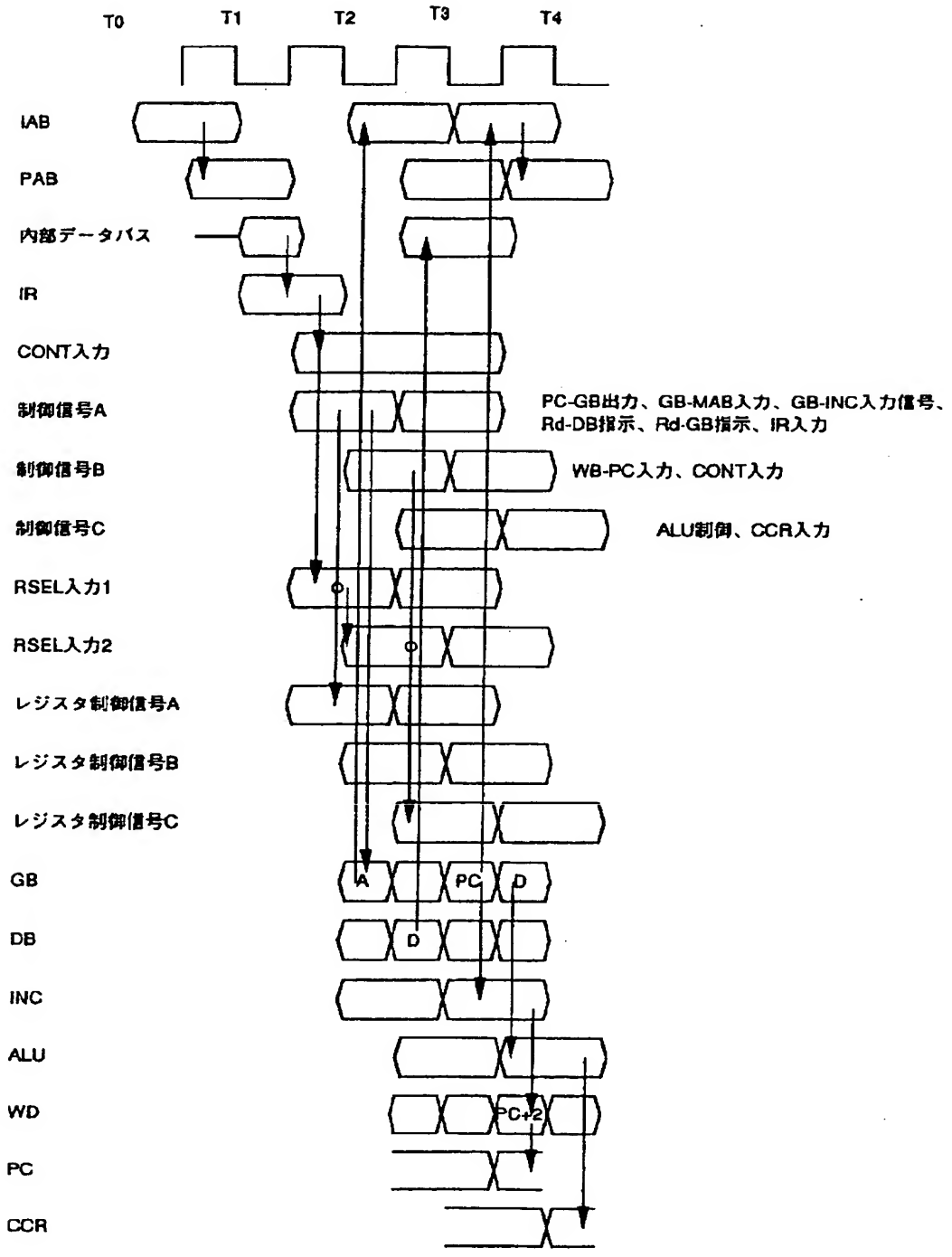
【図５０】

図５０



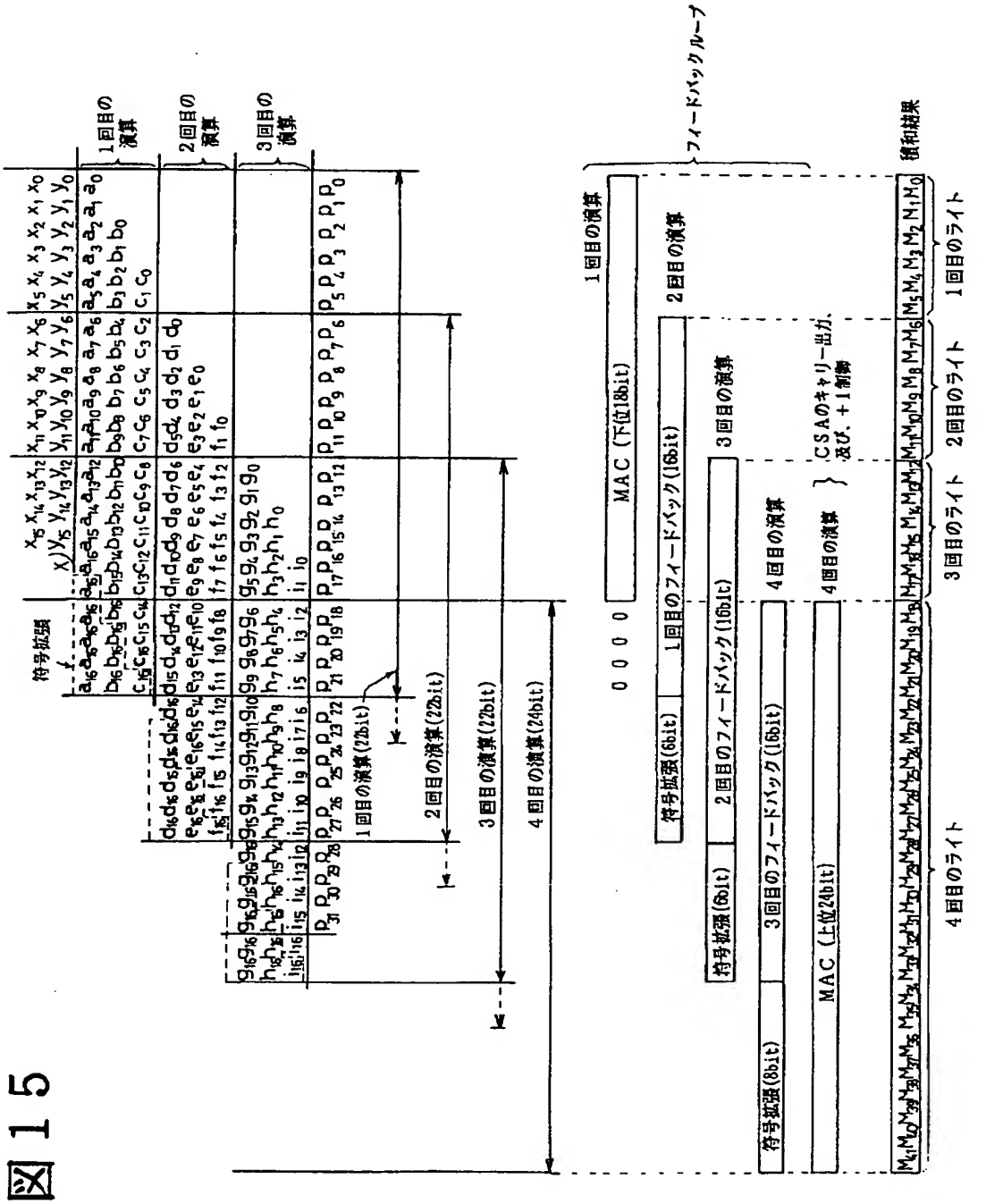
【図１３】

図１３


T0
T1
T2
T3
T4
IAB
PAB
内部データバス
IR
CONT入力
制御信号A
制御信号B
制御信号C
RSEL入力1
RSEL入力2
レジスタ制御信号A
レジスタ制御信号B
レジスタ制御信号C
GB
DB
INC
ALU
WD
PC
CCR
PC-GB出力、GB-MAB入力、GB-INC入力信号、
Rd-DB指示、Rd-GB指示、IR入力
WB-PC入力、CONT入力
ALU制御、CCR入力
A
PC
D
D
PC+2

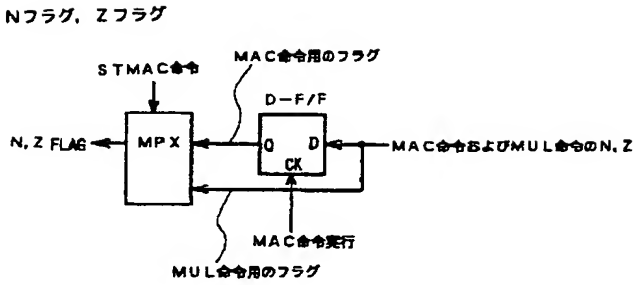
【図１５】

図１５


符号拡張
1回目の演算
2回目の演算
3回目の演算
1回目の演算(22bit)
2回目の演算(22bit)
3回目の演算(22bit)
4回目の演算(24bit)
0 0 0 0
1回目の演算
MAC (下位18bit)
フィードバックループ
符号拡張(6bit)
1回目のフィードバック(16bit)
2回目の演算
符号拡張(6bit)
2回目のフィードバック(16bit)
3回目の演算
符号拡張(8bit)
3回目のフィードバック(16bit)
4回目の演算
MAC (上位24bit)
4回目の演算
CSAのキャリー出力
及び、+1制御
積和結果
4回目のライト
3回目のライト
2回目のライト
1回目のライト

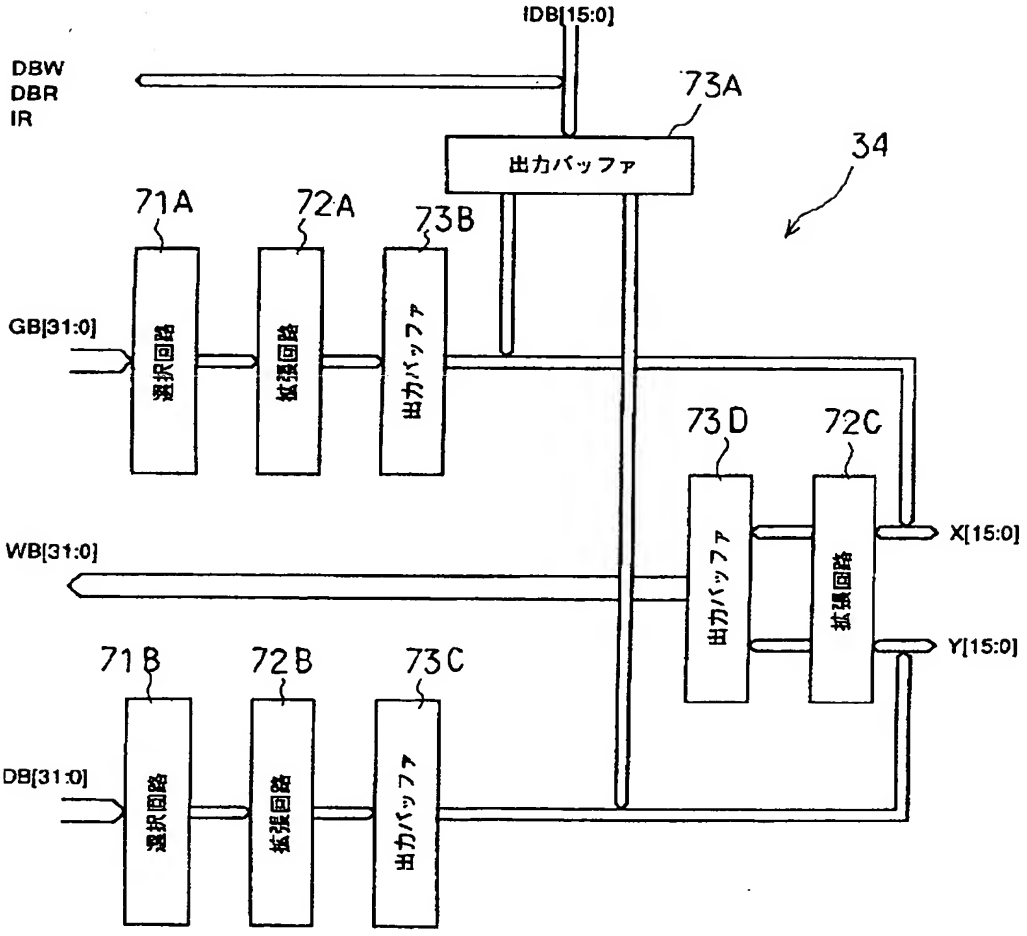
【図１７】

図１７



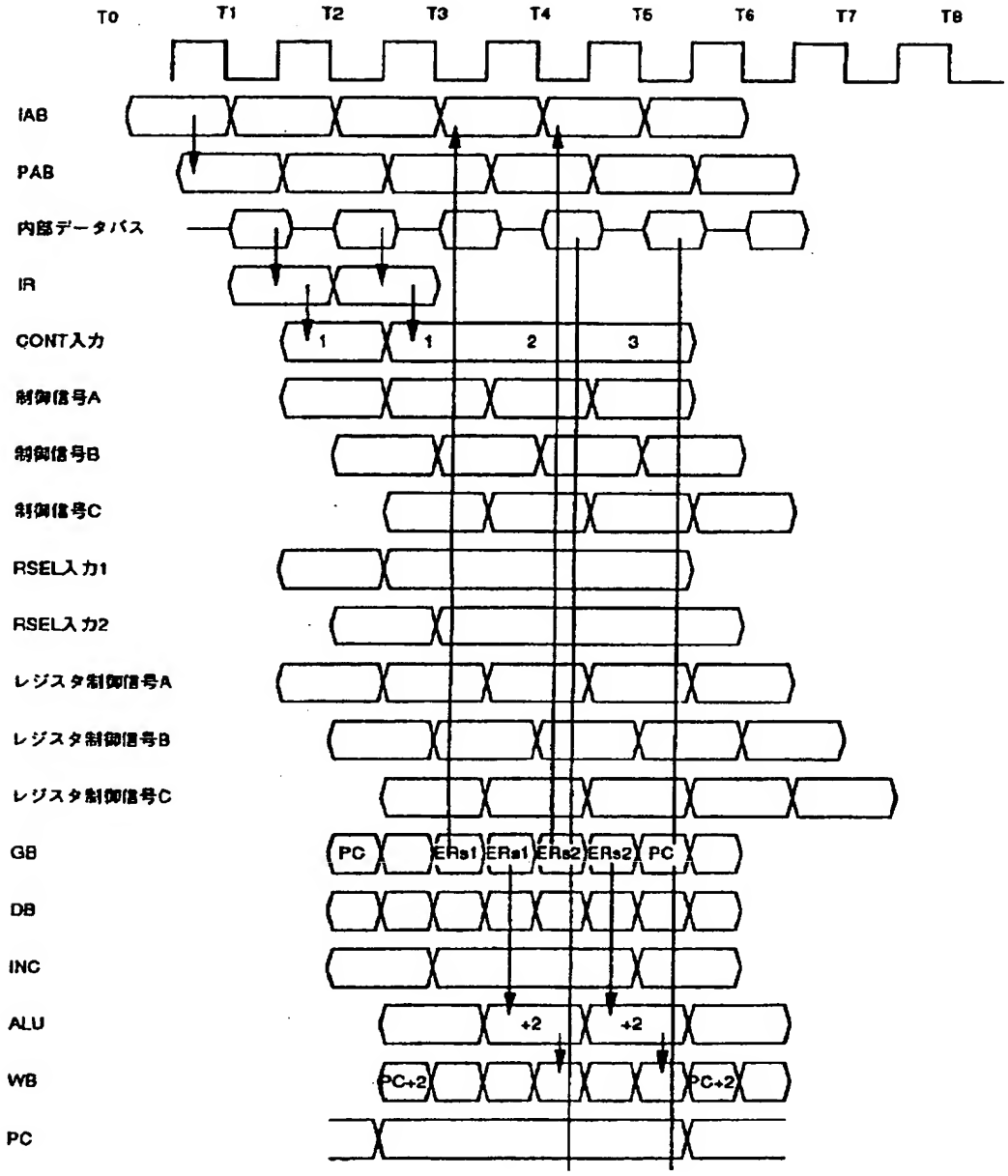
【図１８】

図１８



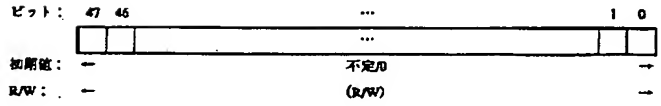
【図１９】

図１９



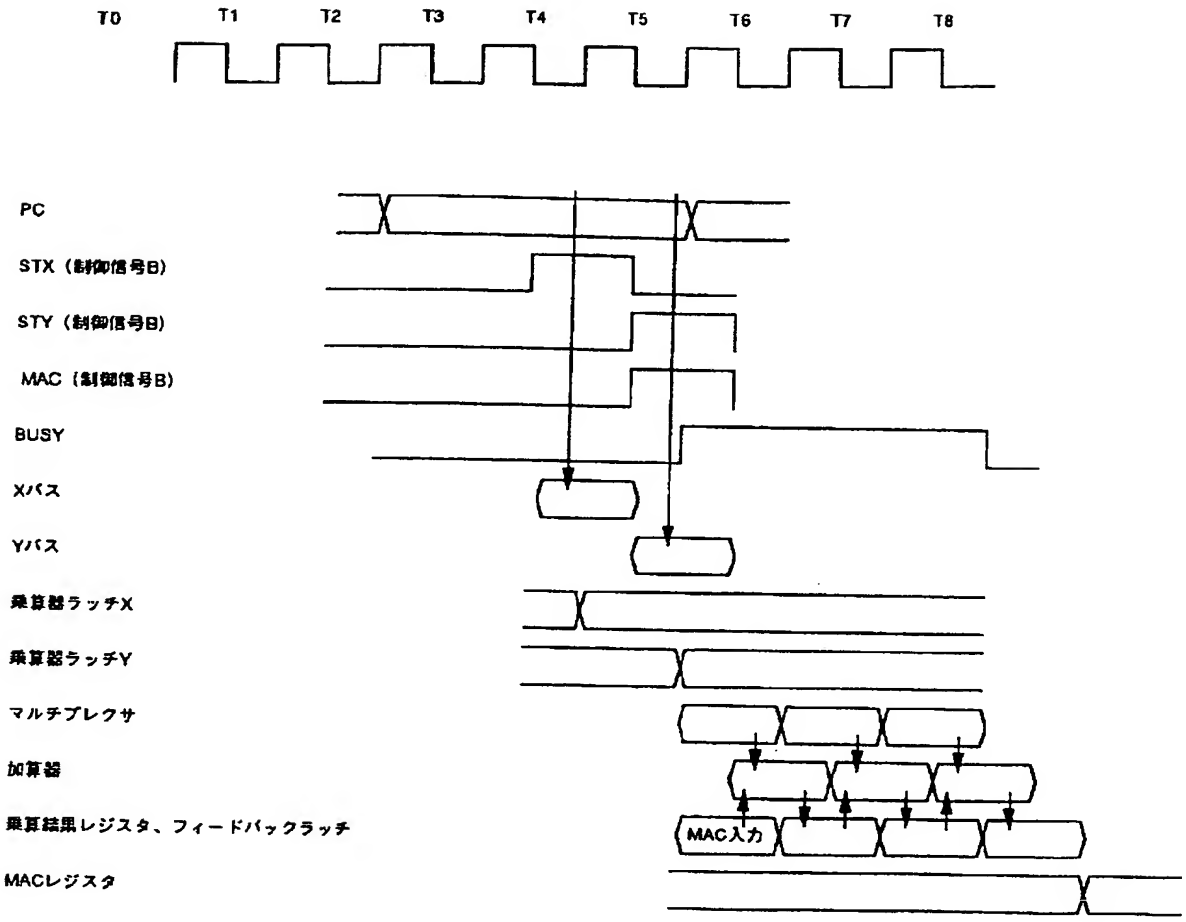
【図４９】

図４９



【図２０】

図２０



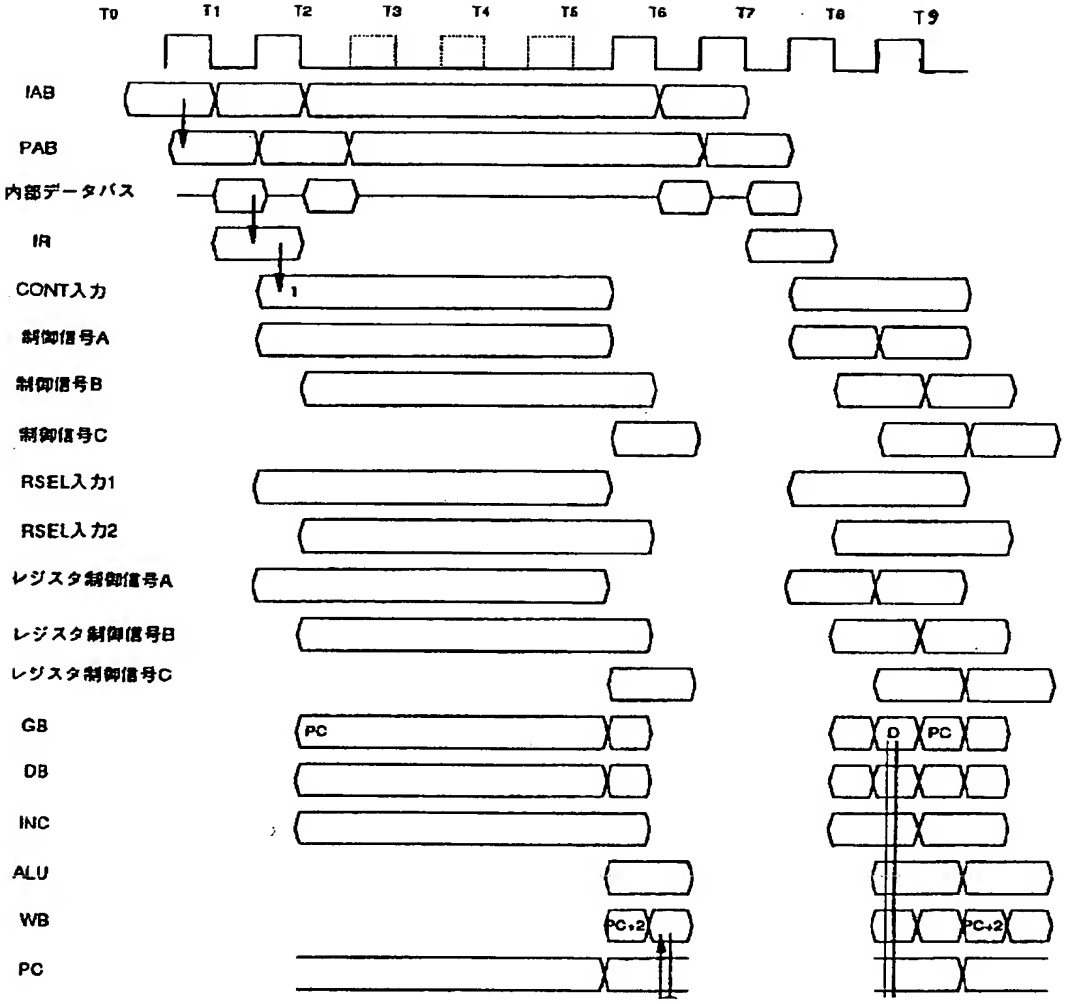
【図３４】

図３４

命令	RSEL2 入力初期値	2回目動作	3回目動作	4回目動作
2本STM	**0	STM制御信号0=1		
3本STM	*00	STM制御信号0=1	STM制御信号1=1	
4本STM	*00	STM制御信号0=1	STM制御信号1=1	STM制御信号1、0=1
2本LDM	**1	LDM制御信号0=1		
3本LDM	*10	LDM制御信号1=1 STM制御信号1=1	LDM制御信号1=1	
4本LDM	*11	LDM制御信号0=1	LDM制御信号1=1	LDM制御信号1、0=1

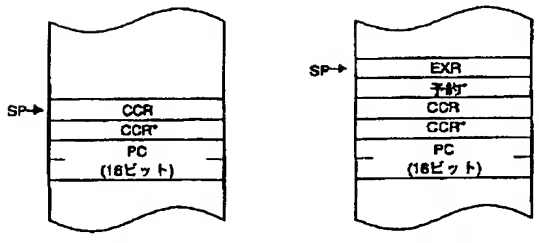
【図２１】

図２１



【図４０】

図４０



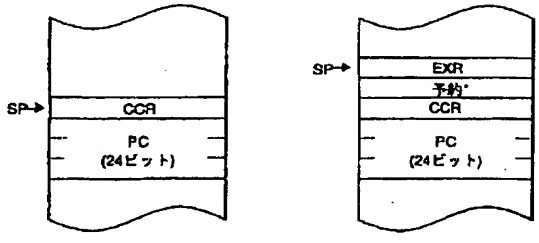
(a)割込み制御モード0、1　　(b)割込み制御モード2、3

* リターン時には無視されます。

例外処理終了後のスタックの状態（ノーマルモード）

【図４１】

図４１



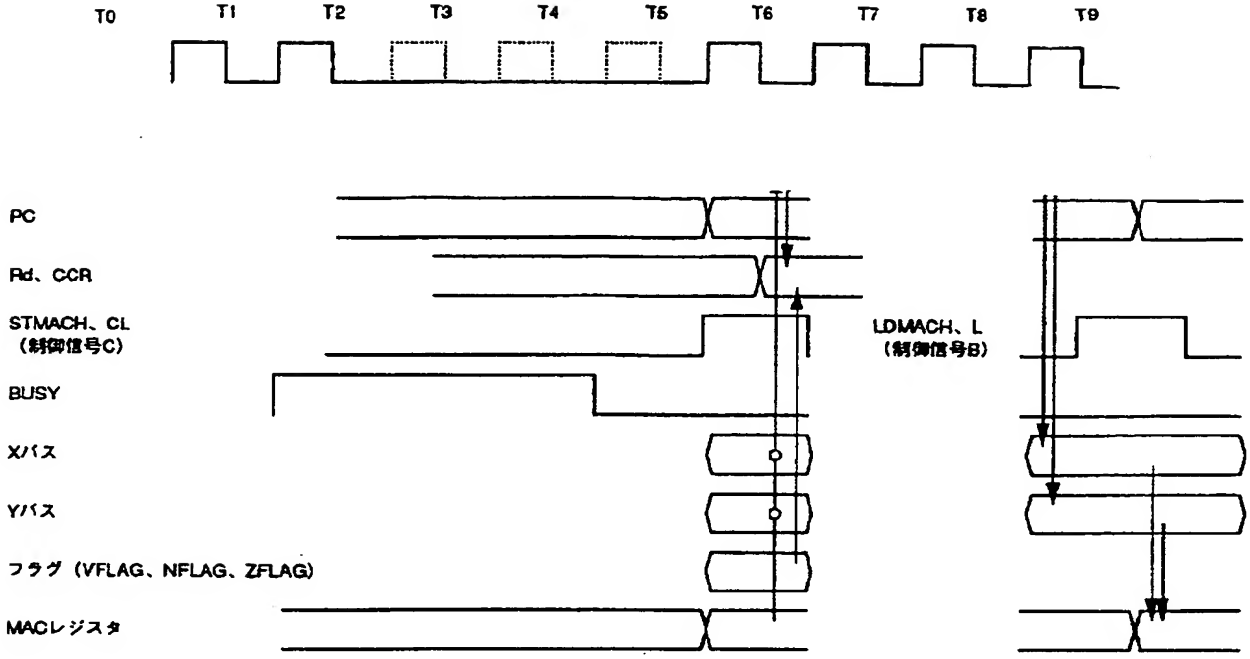
(a)割込み制御モード0、1　　(b)割込み制御モード2、3

* リターン時には無視されます。

例外処理終了後のスタックの状態（アドバンストモード）

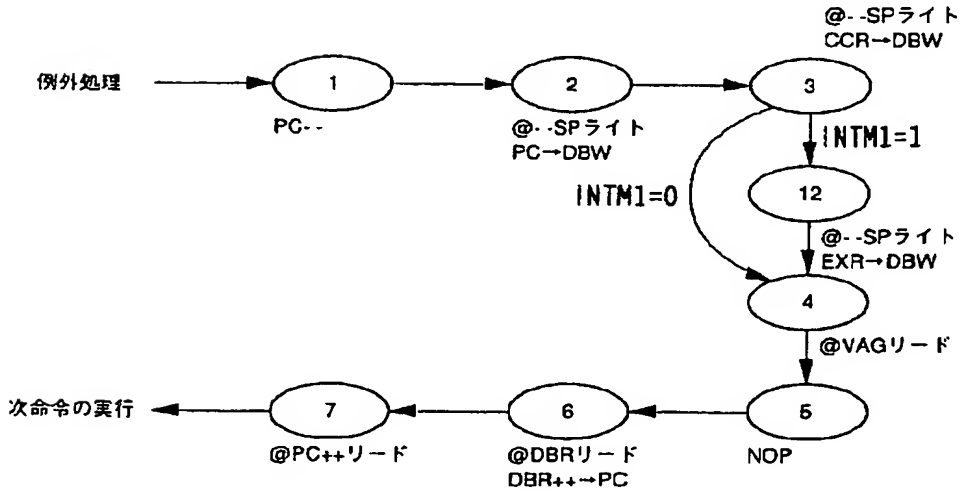
【図２２】

図２２



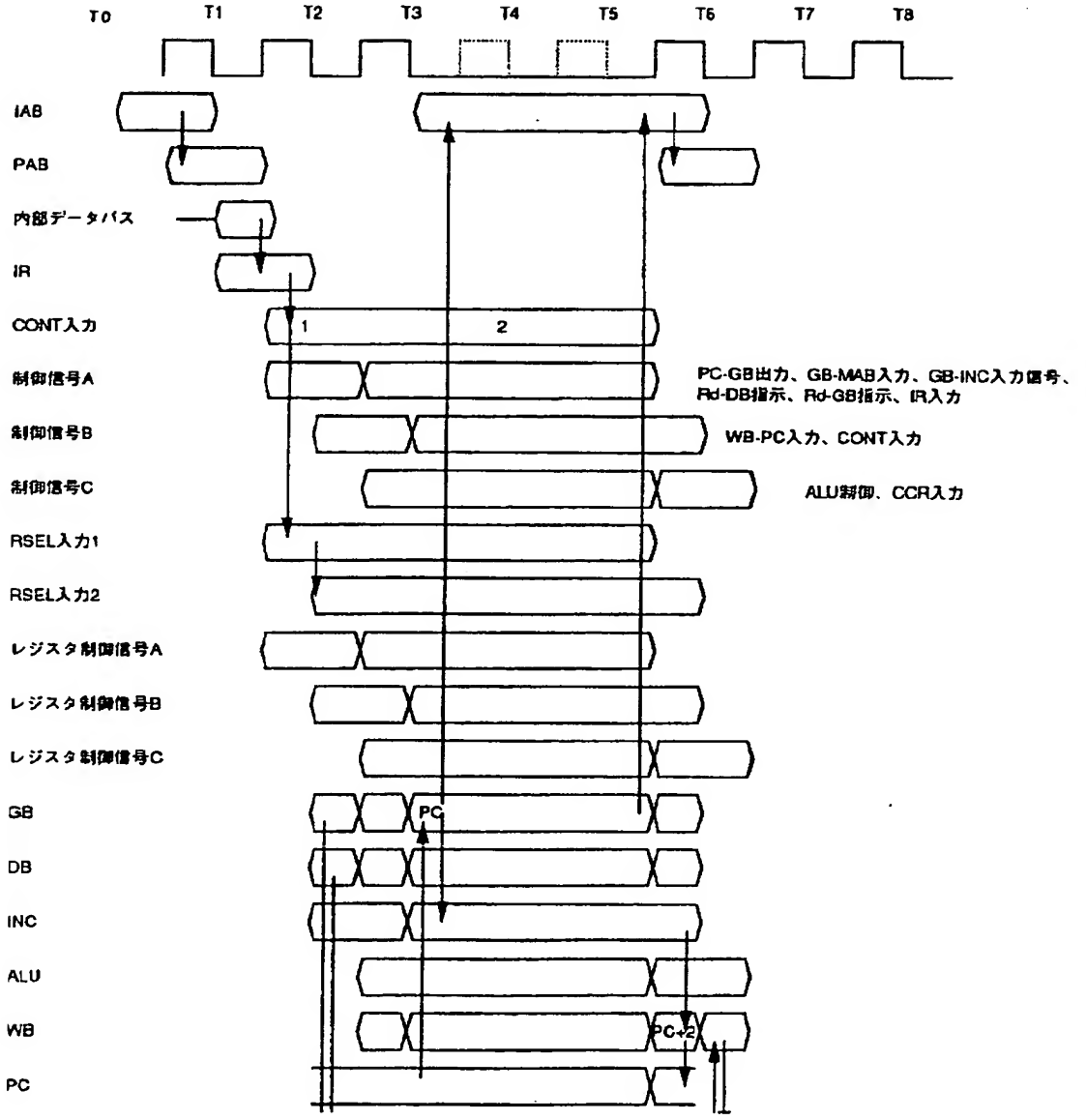
【図３９】

図３９



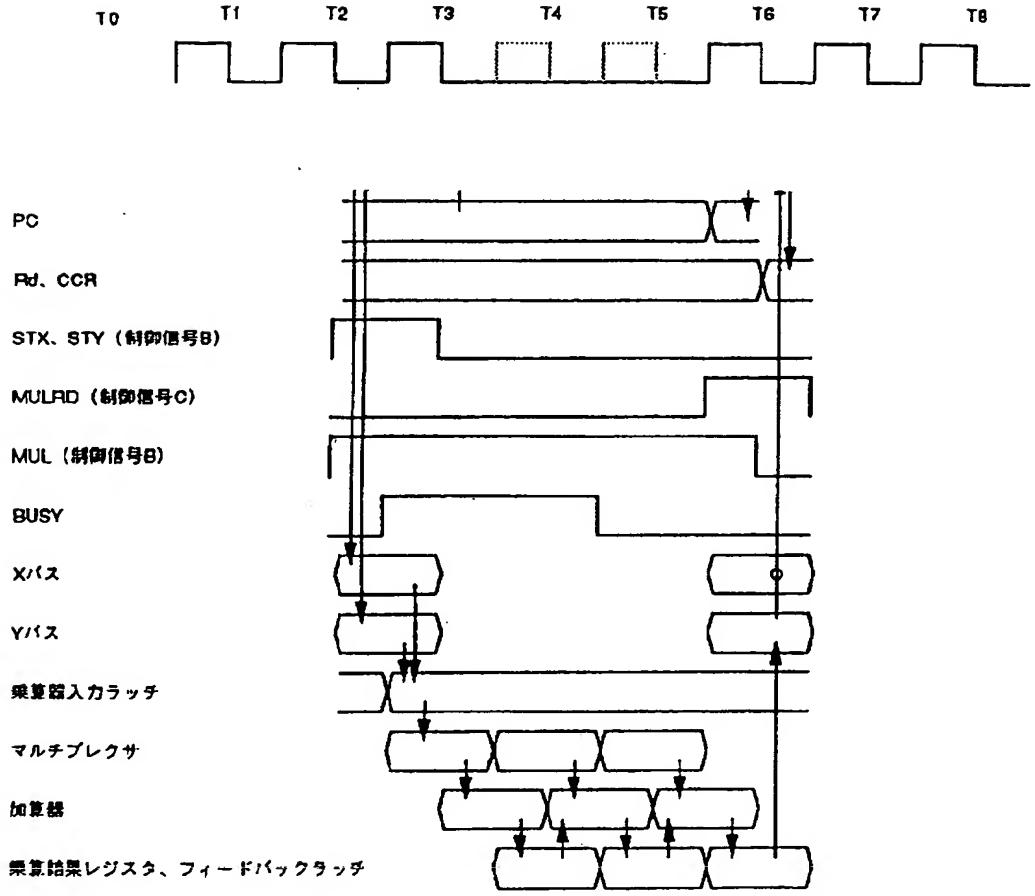
【図２３】

図２３


T0
T1
T2
T3
T4
T5
T6
T7
T8
IAB
PAB
内部データバス
IR
CONT入力
制御信号A
制御信号B
制御信号C
RSEL入力1
RSEL入力2
レジスタ制御信号A
レジスタ制御信号B
レジスタ制御信号C
GB
DB
INC
ALU
WB
PC
1
2
PC-GB出力、GB-MAB入力、GB-INC入力信号、
Rd-DB指示、Rd-GB指示、IR入力
WB-PC入力、CONT入力
ALU制御、CCR入力
PC
PC+2

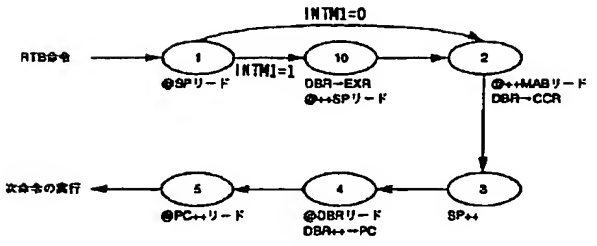
【図２４】

図２４



【図４３】

図４３



【図４６】

図４６

(a)BRCRA

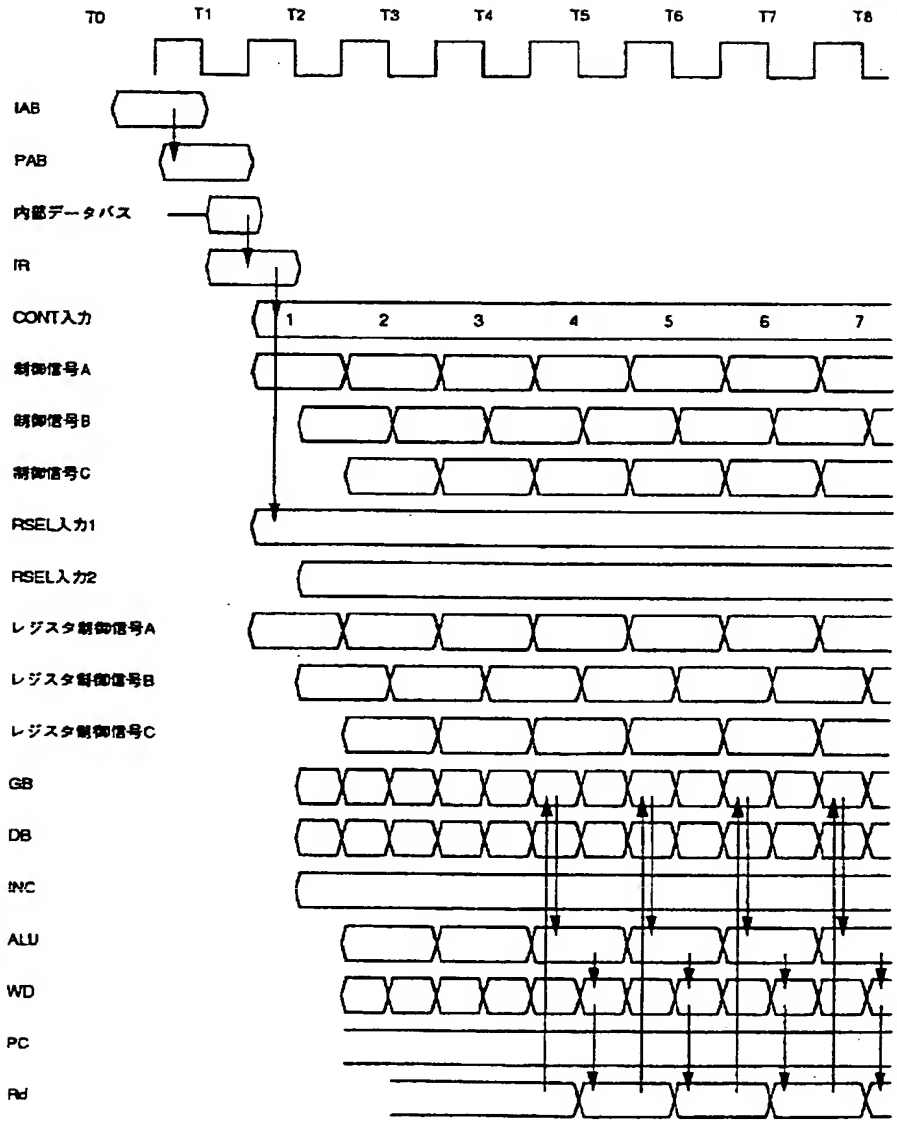
ビット：	7	6	5	4	3	2	1	0
	CMFA	-	TOEA	-	-	CSELA1	CSELA0	BIEA
初期値：	不定/0	不定	不定/0	不定	不定	不定/0	不定/0	不定/0
R/W：	(R/W)	-	(R/W)	-	-	(R/W)	(R/W)	(R/W)

(b)BRCRB

ビット：	7	6	5	4	3	2	1	0
	CMFB	-	TOEB	-	-	CSELB1	CSELB0	BIEB
初期値：	不定/0	不定	不定/0	不定	不定	不定/0	不定/0	不定/0
R/W：	(R/W)	-	(R/W)	-	-	(R/W)	(R/W)	(R/W)

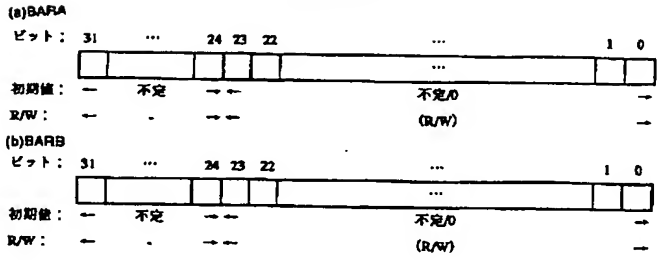
【図２５】

図２５



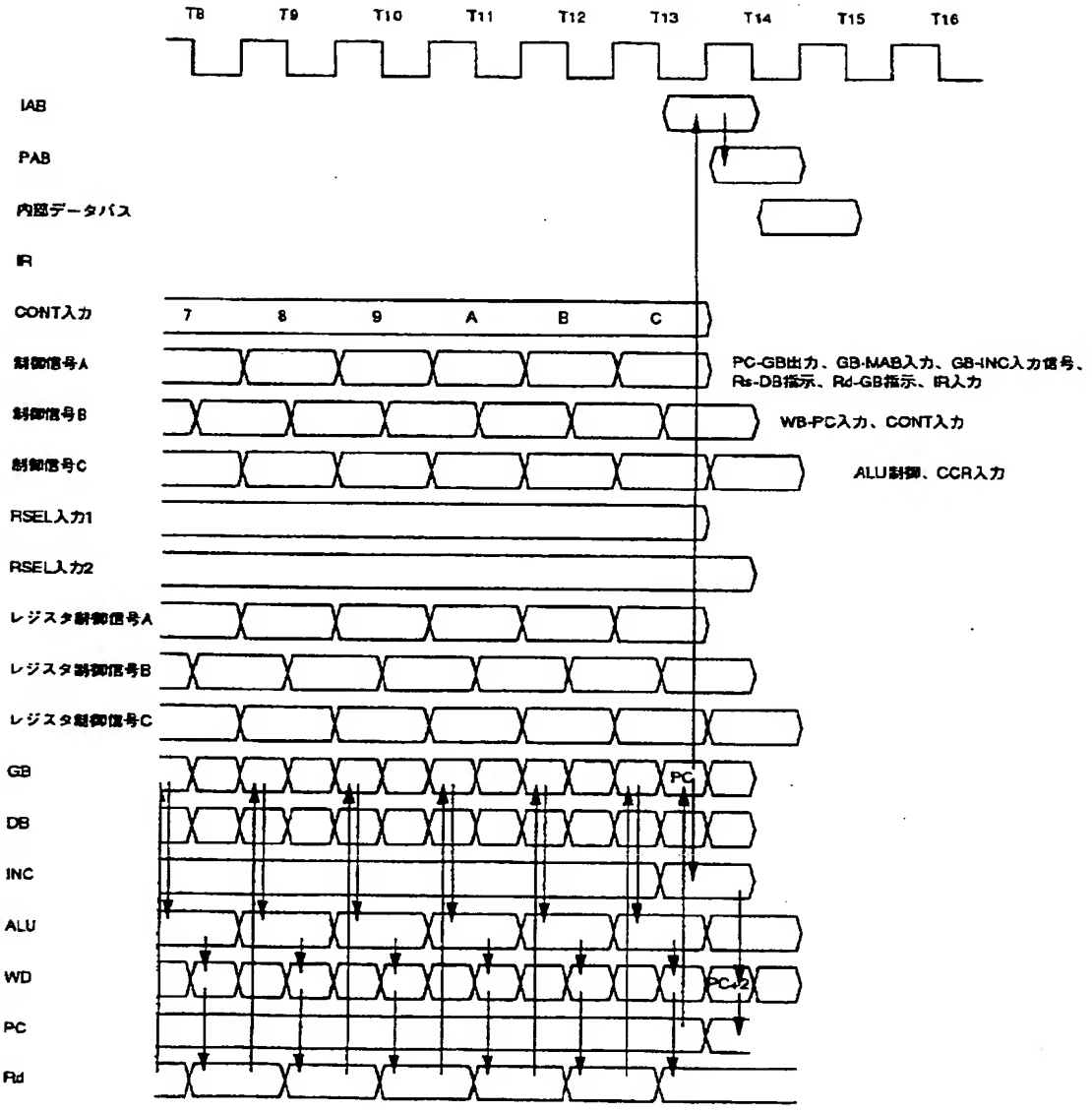
【図４７】

図４７



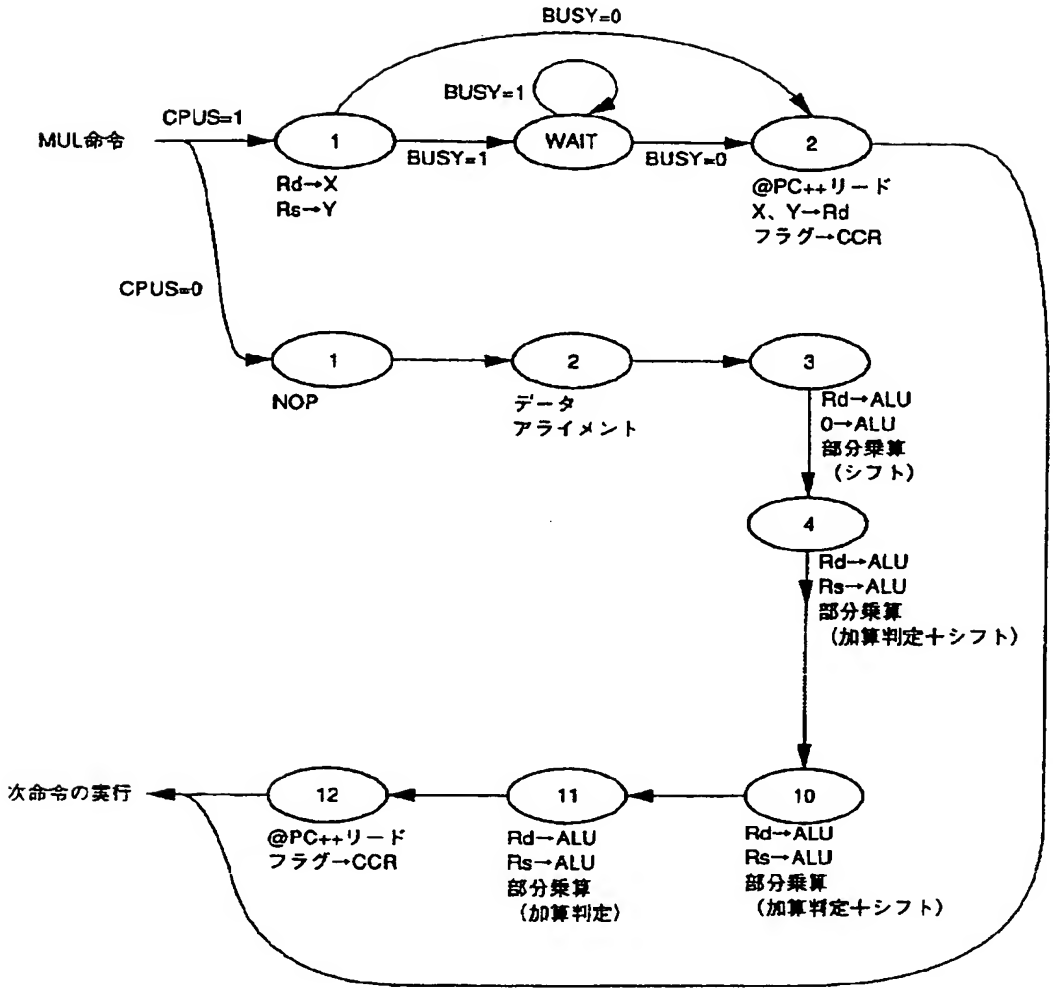
【図２６】

図２６


T8
T9
T10
T11
T12
T13
T14
T15
T16
IAB
PAB
内部データバス
IR
CONT入力
7
8
9
A
B
C
制御信号A
PC-GB出力、GB-MAB入力、GB-INC入力信号、
Rs-DB指示、Rd-GB指示、IR入力
制御信号B
WB-PC入力、CONT入力
制御信号C
ALU制御、CCR入力
RSEL入力1
RSEL入力2
レジスタ制御信号A
レジスタ制御信号B
レジスタ制御信号C
GB
DB
INC
ALU
WD
PC
Rd
PC
PC+2

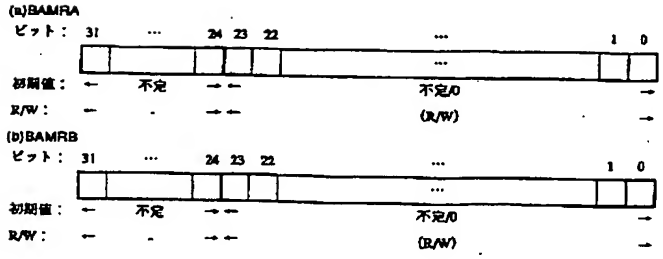
【図２７】

図２７



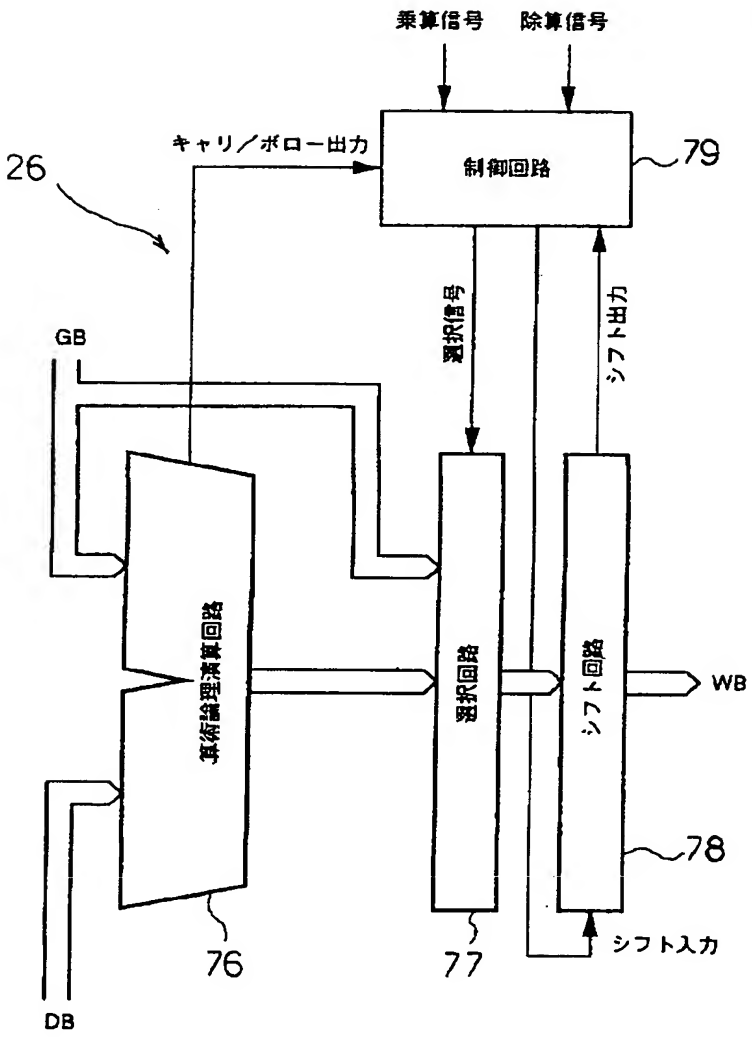
【図４８】

図４８



【図２８】

図２８



【図３６】

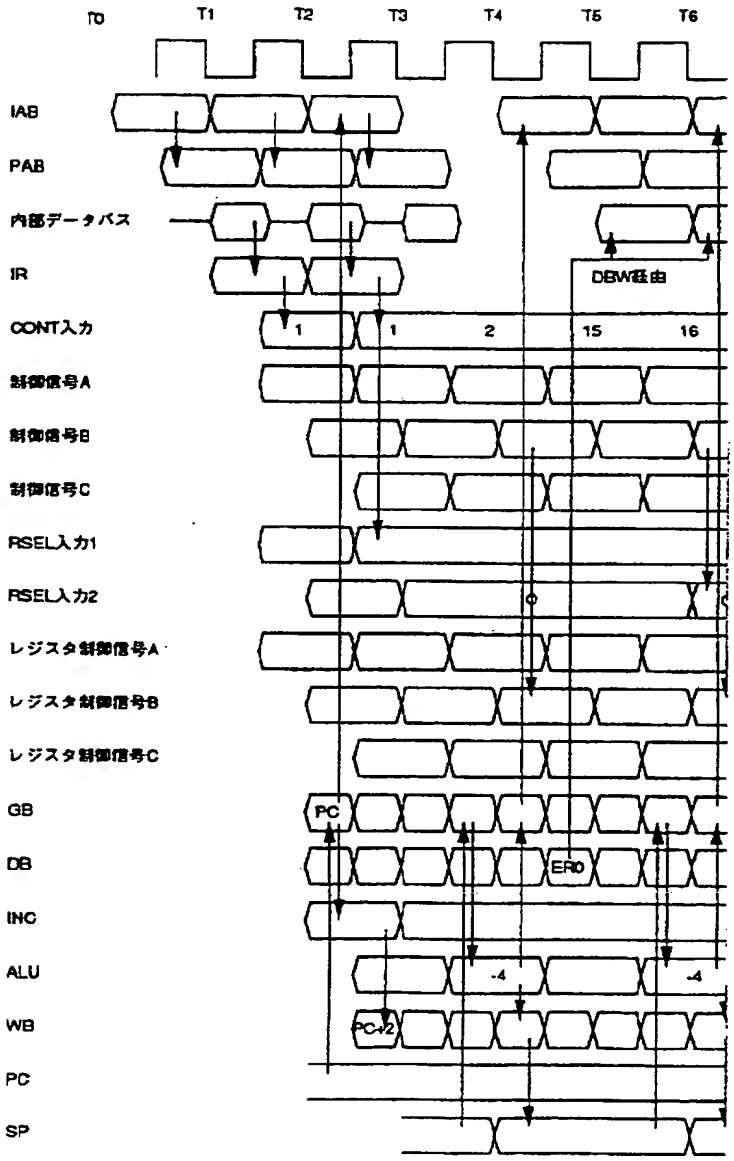
図３６

```
                270:    #undef NextRecord
                271:    }
00000290 01206D76               LDM.L       @SP+,(ER4-ER6)
00000294 01106D73               LDM.L       @SP+,(ER2-ER3)
00000298 5470                   RTS
                272:

                 }
                 |
                297:
                298:    void Proc3(register RecordPtr *PtrParOut)
000002C4                _Proc3:                         ; function: Proc3
000002C4 01006DF5               PUSH.L      ER5
                299:    {
000002C8 7A01                   MOV.L       #_PtrGlb:32,ER1
         <00000000>
000002CE 0F85                   MOV.L       ER0,ER5
                300:    #ifdef DEBUG
                301:    printf("Proc3 executed.\n");
                302:    #endif
                303:      if(PtrGlb!=NULL)
000002D0 01006910               MOV.L       @ER1,ER0
000002D4 470E                   BEQ         L190:8
                304:        *PtrParOut=PtrGlb->PtrComp;
000002D6 01006910               MOV.L       @ER1,ER0
000002DA 01006900               MOV.L       @ER0,ER0
000002DE 01006FD0               MOV.L       ER0,@ER5
000002E2 400A                   BRA         L191:8
000002E4            L190:
                305:      else
                306:        IntGlob=100;
000002E4 79000064               MOV.W       #100:16,R0
000002E8 6BA0                   MOV.W       R0,@_IntGlob:32
         <00000000>
000002EE            L191:
000002EE 01006915               MOV.L       @ER1,ER5
000002F2 7A1500000              ADD.L       #6:32,ER5
         06
000002F8 6B20                   MOV.W       @_IntGlob:32,R0
         <00000000>
000002FE 7910000C               ADD.W       #12:16,R0
00000302 69D0                   MOV.W       R0,@ER5
                307:      Proc7(10,IntGlob,&PtrGlb->IntComp);
                308:    }
00000304 01006D75               POP.L       ER5
00000308 5470                   RTS
                309:
                 |
                 |
```

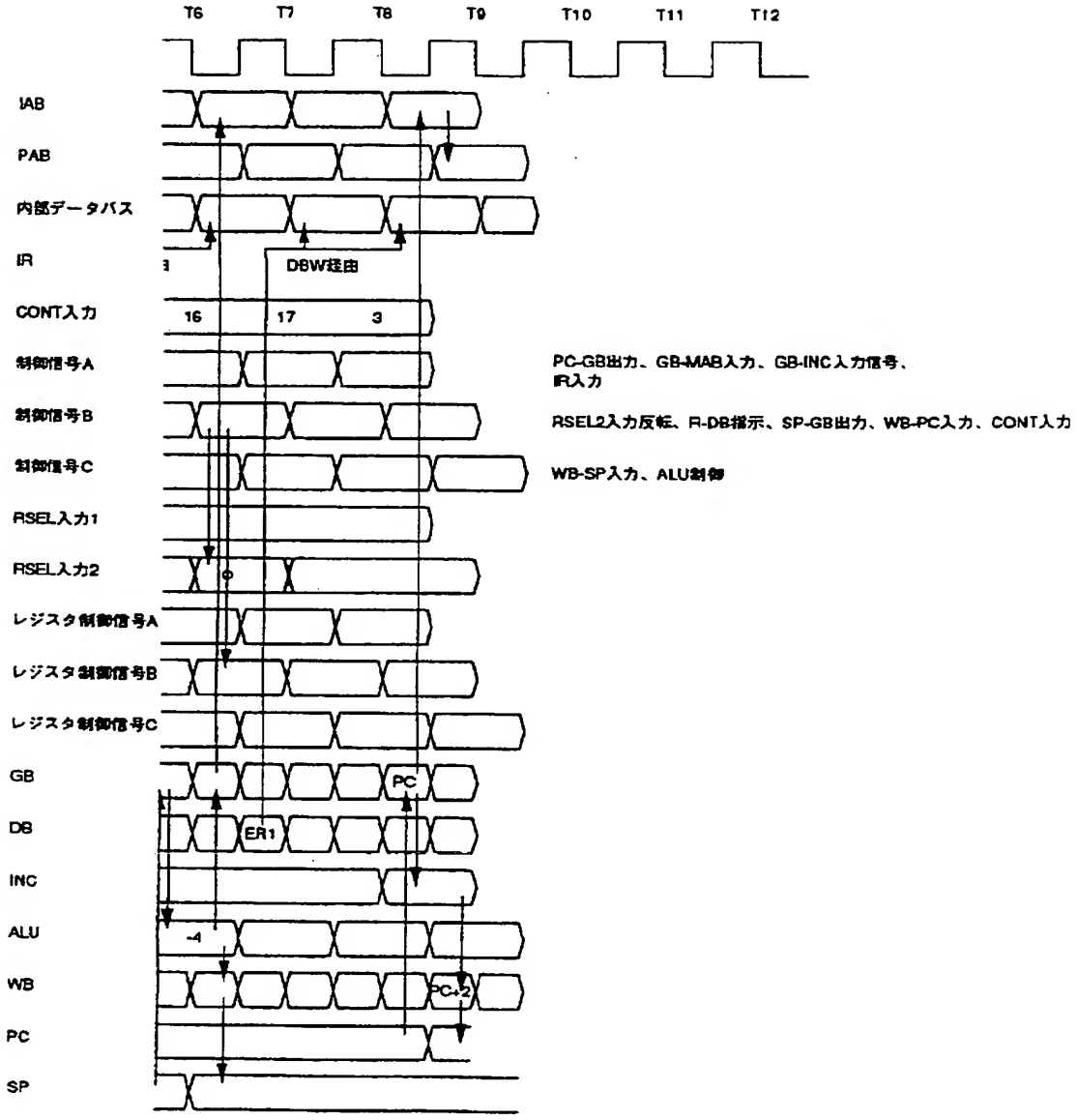
【図２９】

図２９


T0
T1
T2
T3
T4
T5
T6
IAB
PAB
内部データバス
IR
CONT入力
制御信号A
制御信号B
制御信号C
RSEL入力1
RSEL入力2
レジスタ制御信号A
レジスタ制御信号B
レジスタ制御信号C
GB
DB
INC
ALU
WB
PC
SP
DBW経由
1
1
2
15
16
PC
ER0
-4
-4
PC+2

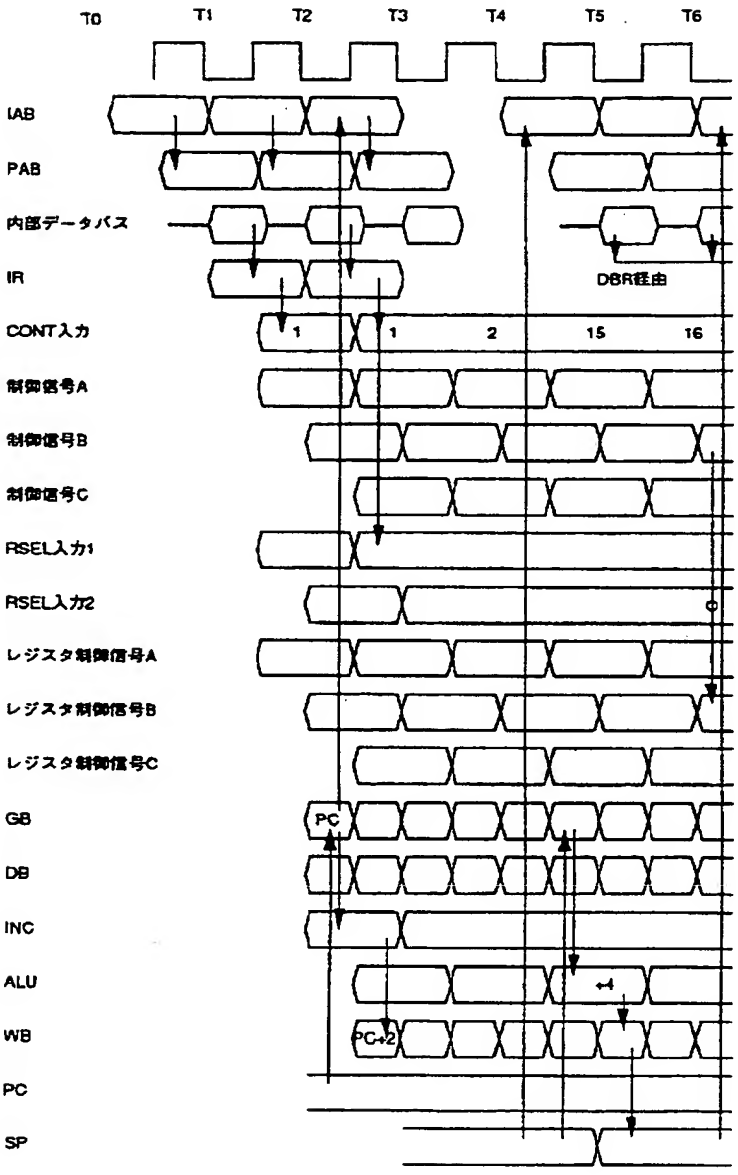
【図３０】

図３０


T6
T7
T8
T9
T10
T11
T12
IAB
PAB
内部データバス
IR
DBW迂回
CONT入力
16
17
3
制御信号A
制御信号B
制御信号C
RSEL入力1
RSEL入力2
レジスタ制御信号A
レジスタ制御信号B
レジスタ制御信号C
GB
DB
INC
ALU
WB
PC
SP
PC
ER1
-4
PC+2
PC-GB出力、GB-MAB入力、GB-INC入力信号、
IR入力
RSEL2入力反転、R-DB指示、SP-GB出力、WB-PC入力、CONT入力
WB-SP入力、ALU制御

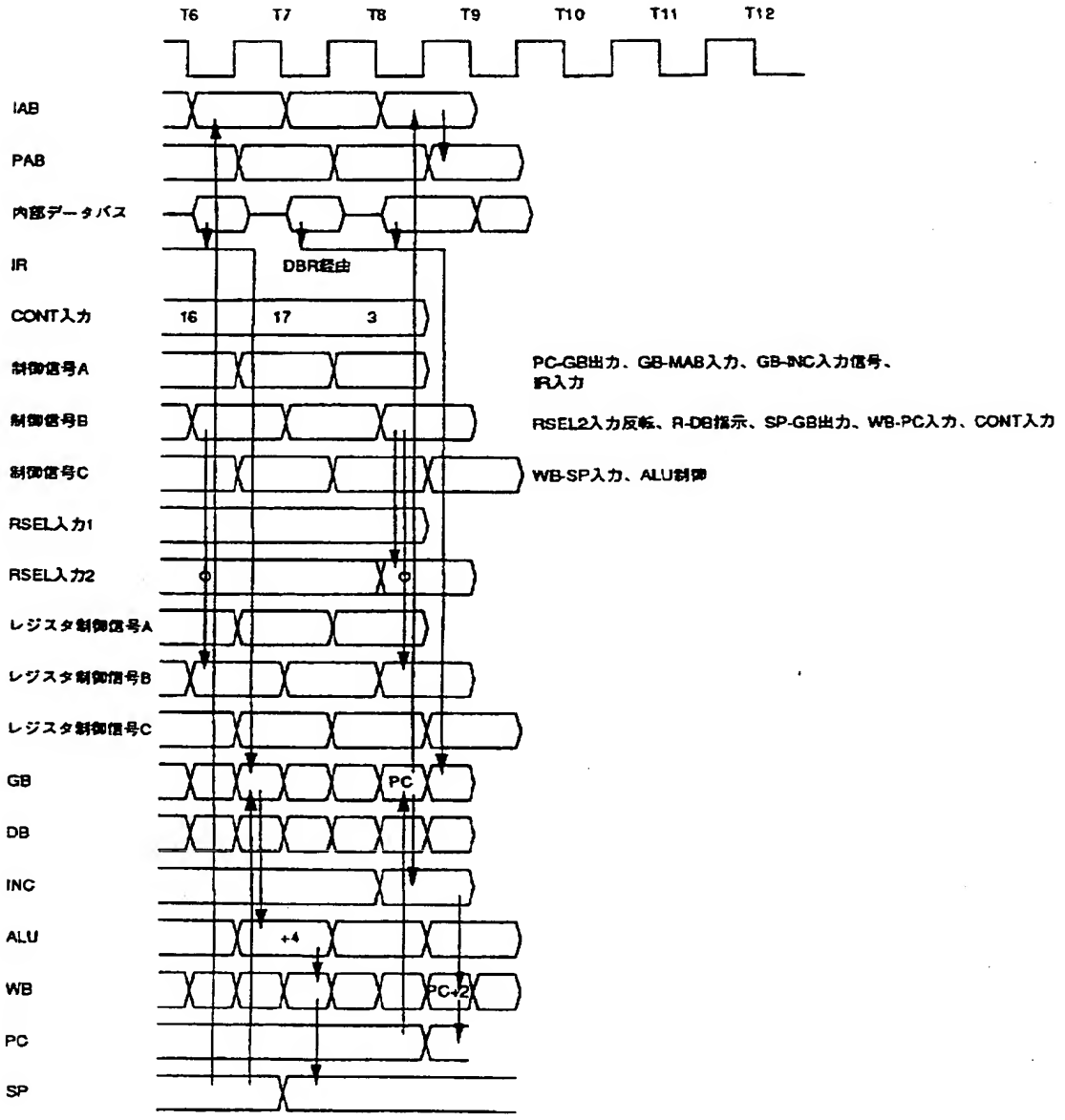
【図３１】

図３１


T0
T1
T2
T3
T4
T5
T6
IAB
PAB
内部データバス
IR
DBR経由
CONT入力
1
1
2
15
16
制御信号A
制御信号B
制御信号C
RSEL入力1
RSEL入力2
レジスタ制御信号A
レジスタ制御信号B
レジスタ制御信号C
GB
PC
DB
INC
ALU
+1
WB
PC+2
PC
SP

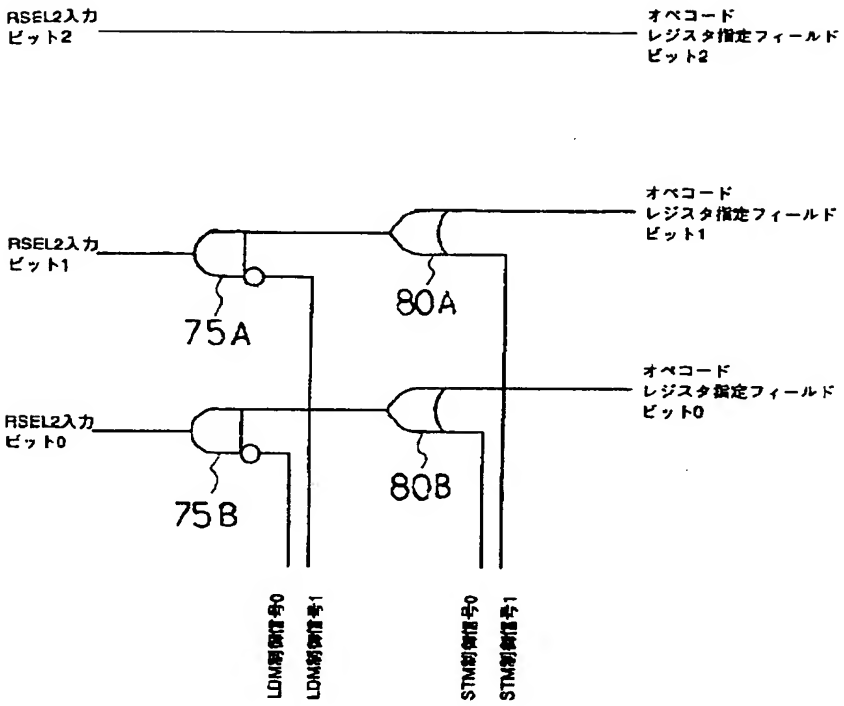
【図３２】

図３２


T6
T7
T8
T9
T10
T11
T12
IAB
PAB
内部データバス
IR
DBR経由
CONT入力
16
17
3
制御信号A
PC-GB出力、GB-MAB入力、GB-INC入力信号、
IR入力
制御信号B
RSEL2入力反転、R-DB指示、SP-GB出力、WB-PC入力、CONT入力
制御信号C
WB-SP入力、ALU制御
RSEL入力1
RSEL入力2
レジスタ制御信号A
レジスタ制御信号B
レジスタ制御信号C
GB
PC
DB
INC
ALU
+4
WB
PC+2
PC
SP

【図３３】

図３３


RSEL2入力
ビット2
オペコード
レジスタ指定フィールド
ビット2
RSEL2入力
ビット1
75A
80A
オペコード
レジスタ指定フィールド
ビット1
RSEL2入力
ビット0
75B
80B
オペコード
レジスタ指定フィールド
ビット0
LDM制御信号0
LDM制御信号1
STM制御信号0
STM制御信号1

【図３５】

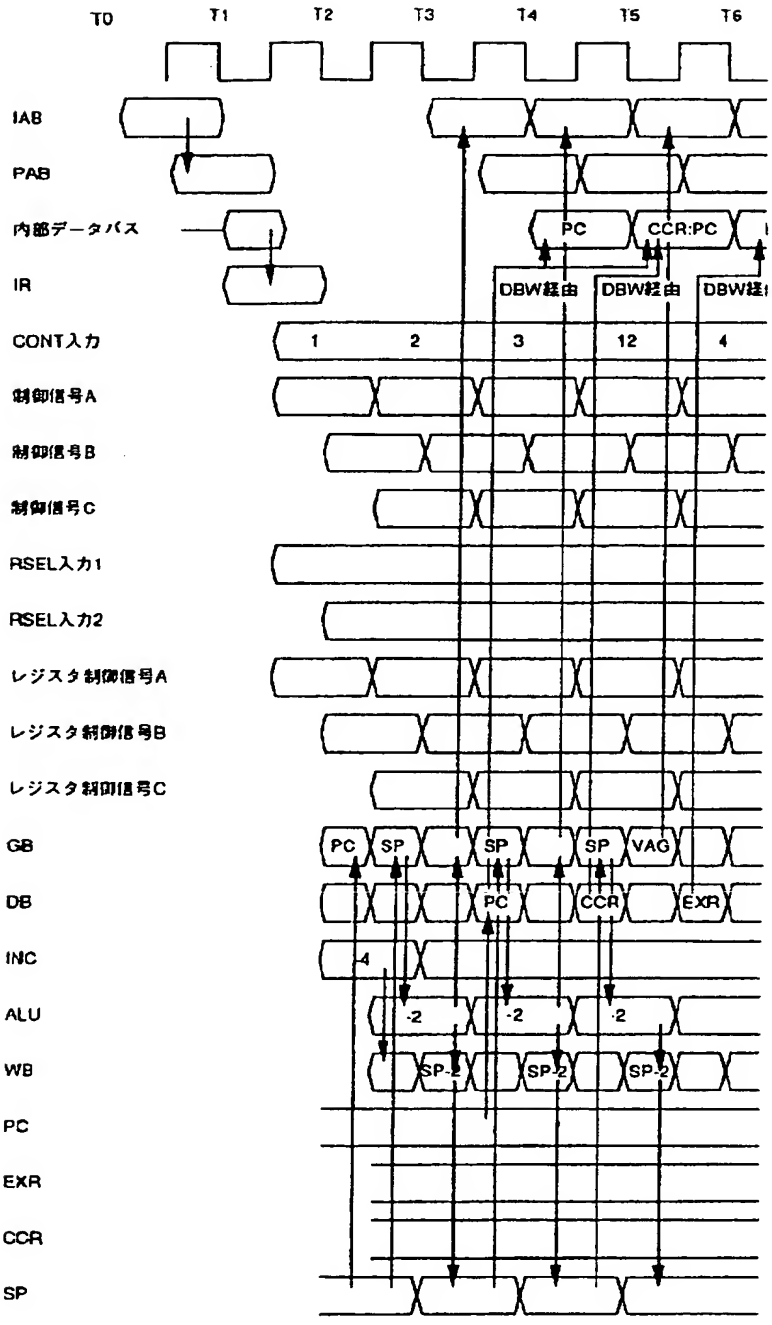
図３５

```
SCT OFFSET CODE       C LABEL     INSTRUCTION OPERAND    COMMENT

                      248:
                      249:    void Proc1(REG RecordPtr PtrParIn)
    000001D6                _Proc1:                              ; function: Proc1
    000001D6 01106DFZ                   STM.L       (ER2-ER3),@-SP
    000001DA 01Z06DF4                   STM.L       (ER4-ER6),@-SP
                      250:    {
    000001DE 0F56                       MOV.L       ER0,ER6
                      251:    #define     NextRecord (*(PtrParIn->PtrComp))
                      252:
                      253:    #ifdef DEBUG
                      254:    printf("Proc1 executed.¥n");
                      255:    #endif
                      256:      structassign(NextRecord),(*PtrGlb));
    000001E0 01006961                   MOV.L       @ER6,ER1
    000001E4 01006B20                   MOV.L       @_PtrGlb:32,ER0
             <00000000>
    000001EC 7A02000000                 MOV.L       #10:32,ER2
             0A
    000001F2                L181:
    000001F2 01006D03                   MOV.L       @ER0+,ER3
    000001F6 01006993                   MOV.L       ER3,@ER1
    000001FA 0B91                       ADDS.L      #4,ER1
    000001FC 1B72                       DEC.L       #1,ER2
    000001FE 46F2                       BNE         L181:8
                      257:      PtrParIn->IntComp=5;
                      259:      NextRecord.PtrComp=PtrParIn->PtrComp;
    00000210 01006950                   MOV.L       @ER6,ER0
    00000214 01006980                   MOV.L       ER0,@ER0
                      260:      Proc3(&NextRecord.PtrComp);
    00000218 01006960                   MOV.L       @ER6,ER0
    0000021C 5C0000A4                   BSR         _Proc3:16
                      261:      if(NextRecord.Discr==Ident1)
    00000220 01006960                   MOV.L       @ER6,ER0
    00000224 6E090004                   MOV.B       @(4:16,ER0),R1L
    00000228 464C                       BNE         L182:8
                      262:      {
```

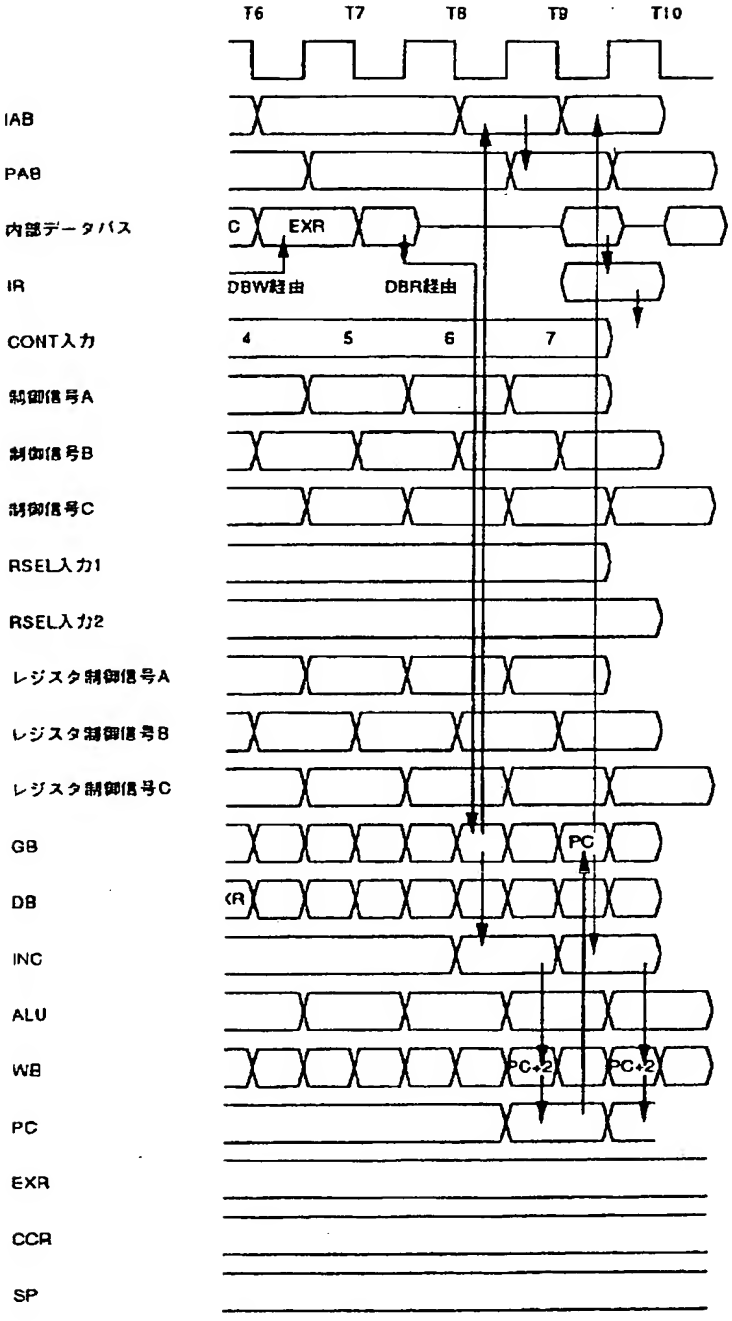
【図３７】

図３７


T0
T1
T2
T3
T4
T5
T6
IAB
PAB
内部データバス
IR
CONT入力
制御信号A
制御信号B
制御信号C
RSEL入力1
RSEL入力2
レジスタ制御信号A
レジスタ制御信号B
レジスタ制御信号C
GB
DB
INC
ALU
WB
PC
EXR
CCR
SP
PC
CCR:PC
DBW経由
DBW経由
1
2
3
12
4
PC
SP
SP
SP
VAG
PC
CCR
EXR
-4
-2
-2
-2
SP-2
SP-2
SP-2

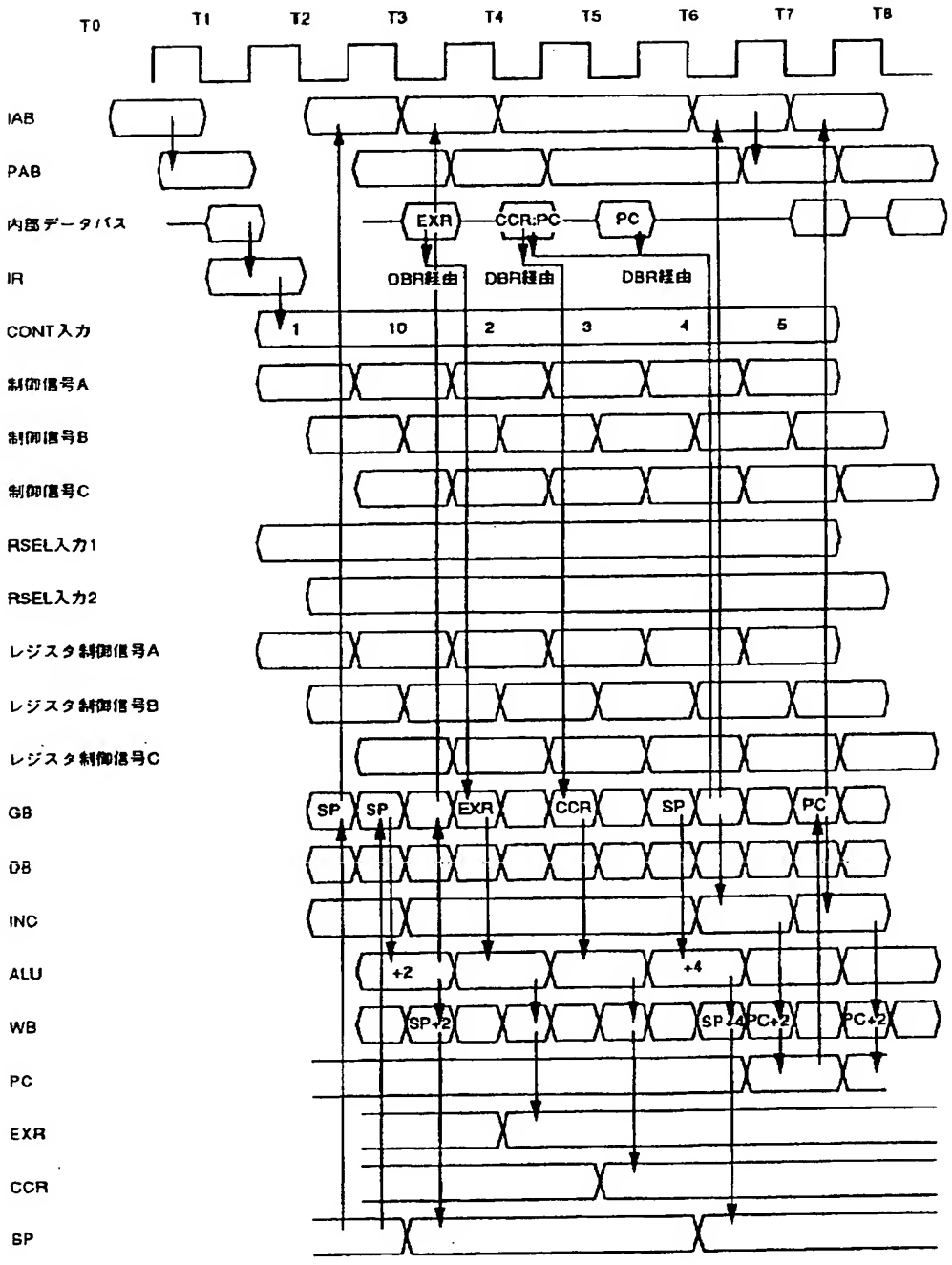
【図３８】

図３８


T6
T7
T8
T9
T10
IAB
PAB
内部データバス
IR
CONT入力
制御信号A
制御信号B
制御信号C
RSEL入力1
RSEL入力2
レジスタ制御信号A
レジスタ制御信号B
レジスタ制御信号C
GB
DB
INC
ALU
WB
PC
EXR
CCR
SP
C
EXR
DBW経由
DBR経由
4
5
6
7
PC
PC+2
PC+2

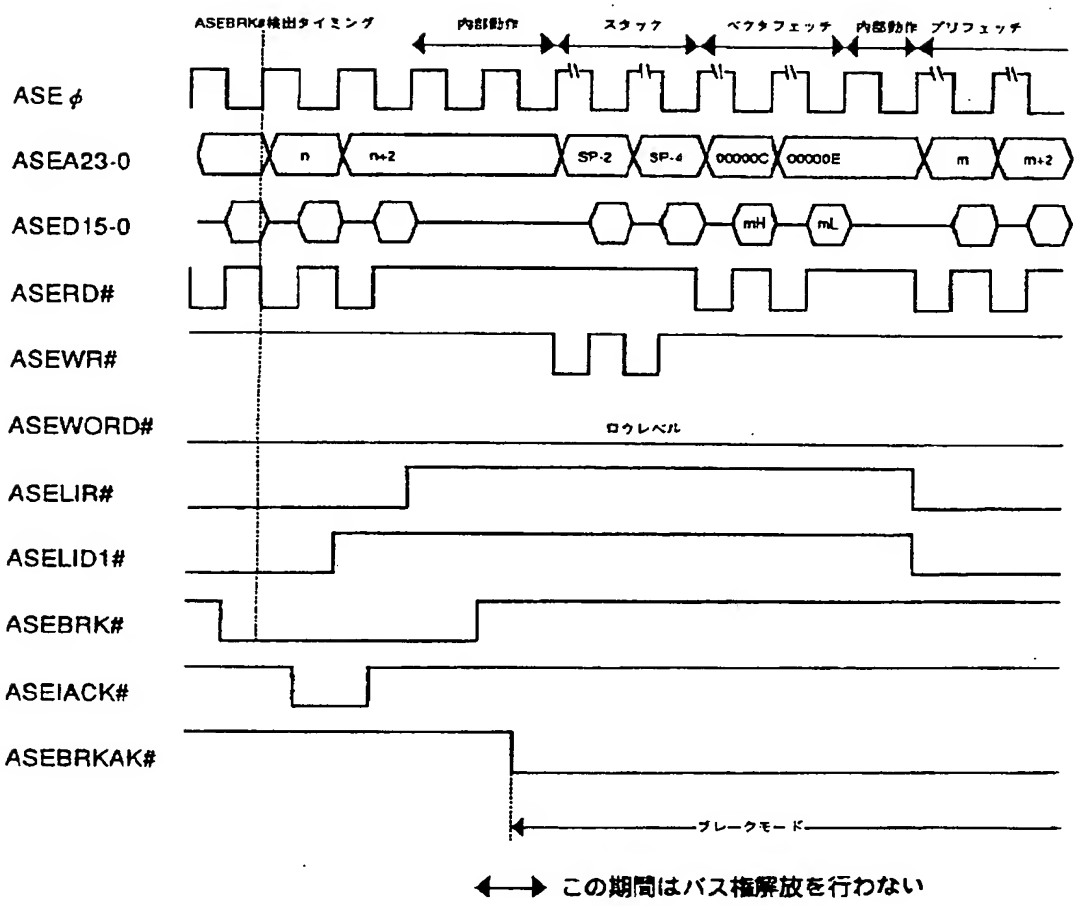
【図４２】

図４２


T0
T1
T2
T3
T4
T5
T6
T7
T8
IAB
PAB
内部データバス
IR
CONT入力
制御信号A
制御信号B
制御信号C
RSEL入力1
RSEL入力2
レジスタ制御信号A
レジスタ制御信号B
レジスタ制御信号C
GB
DB
INC
ALU
WB
PC
EXR
CCR
SP
EXR
CCR:PC
PC
DBR経由
DBR経由
DBR経由
1
10
2
3
4
5
SP
SP
EXR
CCR
SP
PC
+2
+4
SP+2
SP+2
PC+2
PC+2

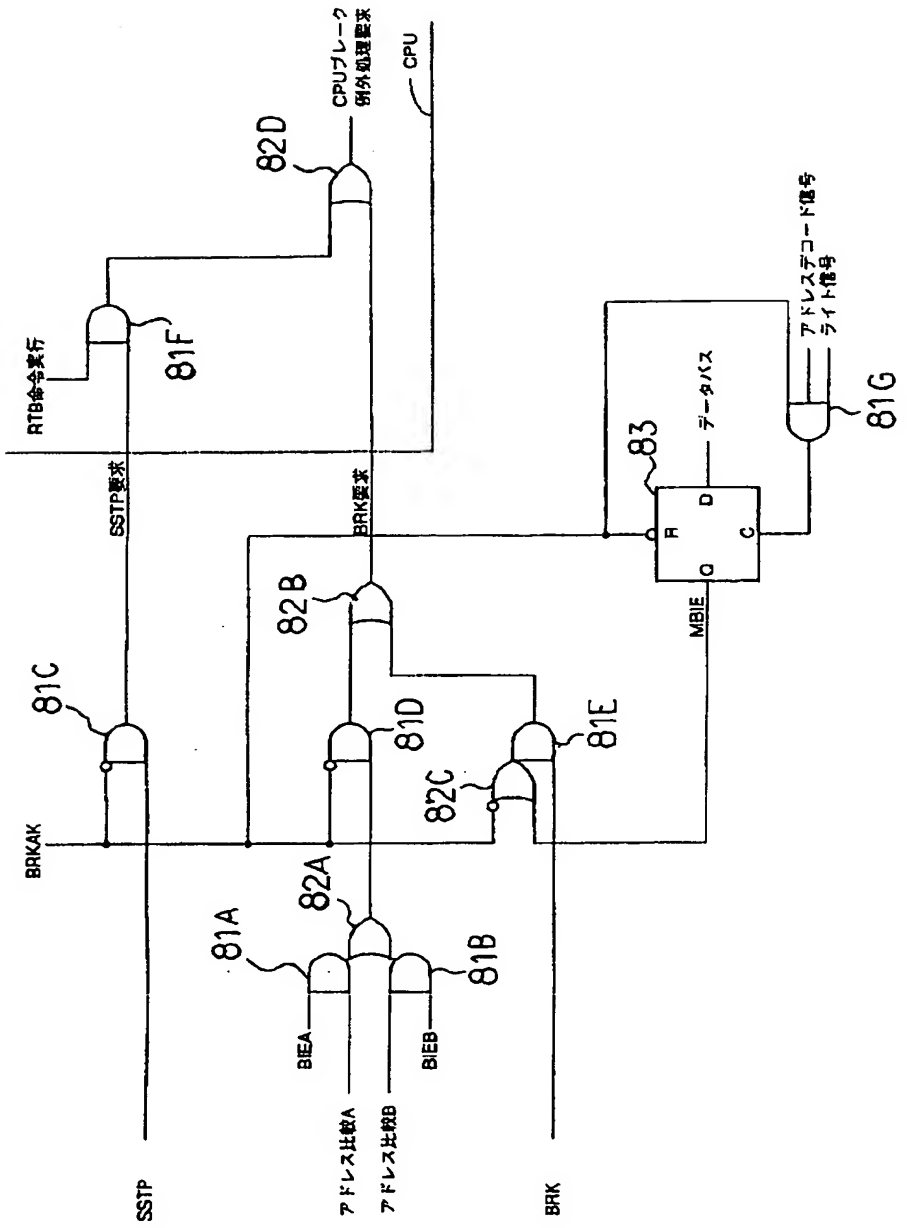
【図５１】

図５１


ASEBRK#検出タイミング
内部動作
スタック
ベクタフェッチ
内部動作
プリフェッチ
ASEφ
ASEA23-0
n
n+2
SP-2
SP-4
00000C
00000E
m
m+2
ASED15-0
mH
mL
ASERD#
ASEWR#
ASEWORD#
ロウレベル
ASELIR#
ASELID1#
ASEBRK#
ASEIACK#
ASEBRKAK#
ブレークモード
この期間はバス権解放を行わない

【図５２】

図５２



フロントページの続き

(72)発明者　梶原　久志
茨城県日立市幸町３丁目２番１号　日立エンジニアリング株式会社内

(72)発明者　宇枝　晋
茨城県日立市幸町３丁目２番１号　日立エンジニアリング株式会社内